Summer 1999

A Family of Hierarchical Encoding Techniques for Image and Video Communications

Samah A. Senbel
Old Dominion University

A FAMILY OF HIERARCHICAL ENCODING TECHNIQUES FOR IMAGE AND VIDEO COMMUNICATIONS

by

Samah A. Senbel
B.Sc. June 1992, Alexandria University, Egypt
M.Sc. December 1994, Alexandria University, Egypt

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY
COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
August 1999

Approved by:

Hussein Abdel-Wahab (Director)

Kurt Maly (Member)

Shunichi Toida (Member)

Christian Wild (Member)

Martin Meyer (Member)

ABSTRACT

A FAMILY OF HIERARCHICAL ENCODING TECHNIQUES FOR IMAGE AND VIDEO COMMUNICATIONS

Samah A. Senbel
Old Dominion University, 1999
Director: Dr. Hussein Abdel-Wahab

As the demand for image and video transmission and interactive multimedia applications continues to grow, scalable image and video compression that has robust behavior over unreliable channels are of increasing interest. These desktop applications require scalability as a main feature due to its heterogeneous nature, since participants in an interactive multimedia application have different needs and processing power. Also, the encoding and decoding algorithm complexity must be low due to the practical considerations of low-cost low-power receiver terminals. This requires image and video encoding techniques that jointly considers compression, scalability, robustness, and simplicity.

In this dissertation, we present a family of image and video-encoding techniques, which are developed to support conferencing applications. We achieve scalability, robustness and low computational complexity by building our encoding techniques based on the quadtree and octree representation methods.

First we developed an image encoding technique using the quadtree representation of images and vector quantization. We use a mean-removal technique to separate the means image and the difference image. The difference image is then encoded as a breadth first traversal of the quadtree corresponding to the image. Vector quantization is then used to compress the quadtree nodes based on the spatial locality of the quadtree data. Our next step was to use the quadtree-based image encoding technique as a base for developing a differential video encoding technique. We extended it to encode video by applying the well-known IPB technique to the

image encoding system.

Then, we explore another method of extending our image encoding technique to encode video streams. The basic idea was to use exactly the same three steps used in our image encoding technique, mean removal, conversion to tree structure, and vector quantization, and replace the quadtree structure with an octree structure. The octree is the three-dimensional equivalent of the quadtree. We divide the sequence of frames into groups and view each group as a three-dimensional object. By encoding frames together, we can obtain substantial savings in encoding time and better compression results.

Finally, we combined both the differential quadtree and octree approaches to generate a new hybrid encoding technique. We encode one frame using the quadtree-based image encoding technique, and then encode the following group of frames as a differential octree based upon the first frame.

Using a set of experiments, the quadtree-based image encoding and differential video encoding techniques were shown to provide reasonable compression in comparison with similar techniques, while the octree and hybrid video encoding techniques gave impressive compression results. Furthermore, we demonstrated that our encoding techniques are time efficient compared to the more common frequency based techniques. We also compare their scalability feature favorably with other well-known scalable techniques. Moreover, we demonstrated their ability to tolerate and conceal error. The new encoding techniques proved to be efficient methods of encoding for interactive multimedia applications.

ACKNOWLEDGMENTS

There are many people who have contributed to the successful completion of this dissertation. My deepest gratitude and appreciation are due to Dr. Hussein Abdel-Wahab, for his persistent direction and constructive guidance throughout this work. I extend my thanks to all of my committee members for their patience and guidance on my research. Also, I would like to thank all the faculty and colleagues at the Computer Science Department of Old Dominion University for their support and encouragement.

I am truly blessed to have a wonderful supportive family. My understanding and encouraging husband, Alaa, my loving son, Ahmed, and my beautiful daughter, Salma, were strong motivations for me to work hard to fulfill this goal. Although miles have often separated me from my parents, they have been a constant source of love and encouragement. I am very thankful to my entire family for their unfailing belief in my ability.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The demand for image and video encoding has greatly increased in the past decade due to factors such as the increased availability of personal workstations, multimedia applications, and networking capabilities. Because the amount of data associated with video is huge, its efficient storage and transmission poses a challenging problem. As a result, there is a lot of research in the field of data compression. Moreover, compression alone is no longer a goal by itself, as different applications need more features. Video conferencing applications have additional requirements: low computational complexity, scalability, and robustness are strongly required features in such systems. In this dissertation, we present our view and efforts in order to design encoding techniques geared towards the particular needs of interactive multimedia applications.

1.1 Overview

Data compression is the process of eliminating or reducing the redundancy in data representation in order to achieve savings in storage and communication costs. For video data, compression is a requirement and not an option, due to the large amount of data. For example, assume that we have a sequence of frames of size 320x240 pixels, a frame rate of 15 frames/second, and simple 8-bit gray scale colors. Each

The journal model for this dissertation is the *IEEE Transactions*.

uncompressed pixel would need 8 bits per frame, so we would need a bandwidth of 9.216 Mbps (320x240x8x15), which is a large bandwidth for a small frame size. A colored video sequence would require three times that bandwidth.

Video encoding techniques can be divided into intra-frame and inter-frame techniques. Intra-frame coding compresses video, image by image, and removes only the spatial redundancy inside each frame. Its advantage is in its simplicity and robustness. Inter-frame coding encodes the sequence of images together as a group, and therefore reduces the temporal redundancy between frames. Its main advantage is achieving high compression results. In our work, we concentrate on inter-frame encoding techniques.

The choice of a compression technique depends on the application environment. The requirements of an interactive application are different from those of a broadcast application. An interactive application requires the ability to change the properties of the video stream dynamically, depending on the participants' needs. A broadcast application typically works blindly independent of the number of receivers and their capacities.

The requirements also differ depending on whether we have a single stream or multiple streams. For example, a video conferencing application that supports several simultaneous participants, may not find enough network bandwidth, or system processing capabilities, to provide a full-motion video stream of each participant. Instead, a participant may receive a full-motion video image of the current speaker, beside low frame rate video images of the rest of the participants.

Also, different environments determine different coding characteristics, such as coding delay, error resilience, scalability, and real-time requirements. For example, in video-telephony, the emphasis is on speed of encoding and compression efficiency, while scalability and robustness are not important. On the other hand, in video-conferencing, the emphasis is on scalability and real-time requirements, while in video databases, the emphasis is on compression efficiency and scalability.

In our work, we concentrate on video conferencing applications and develop

a family of new video encoding techniques tailored for the specific needs of such applications. A good encoding technique for video-conferencing should be robust and dynamic in order to cope with the high demands of interactive video communication.

Existing encoding techniques have some, but not all, of the properties, which are necessary for video-conferencing over the internet. The criticality of building scalability into collaborative multimedia applications has been widely recognized. Many encoding techniques have incorporated scalability as an added feature, but at a cost of extra encoding time. Robustness was rarely considered in a video encoding technique and the application usually relied on the underlying network for error-correction. However, an encoding with error-correction capabilities would be more practical.

In this dissertation, we present a family of lossy, scalable, error-resilient video encoders, with low complexity, which makes them suitable for real-time video conferencing applications. We combine scalability with error protection techniques based on uneven error protection, where important data is favored with more protection. This allows graceful degradation of picture quality as available bandwidth for transmission decreases instead of a sudden breakdown.

1.2 Environment and Applications

Video conferencing has gained a lot of popularity in the last few years due to the increasing growth in network technology. It is now used for distance learning and training, corporate meeting, scientific and engineering co-operative efforts, and Internet games, to mention just a few areas.

Our image and video encoding techniques are designed to work as part of an interactive multimedia application with multiple participants and multiple data streams. A common configuration for multimedia conferences entails small groups of participants connected together by bottleneck links, as well as some participants connected by slow telephone lines to the network. Typically, the bottleneck link

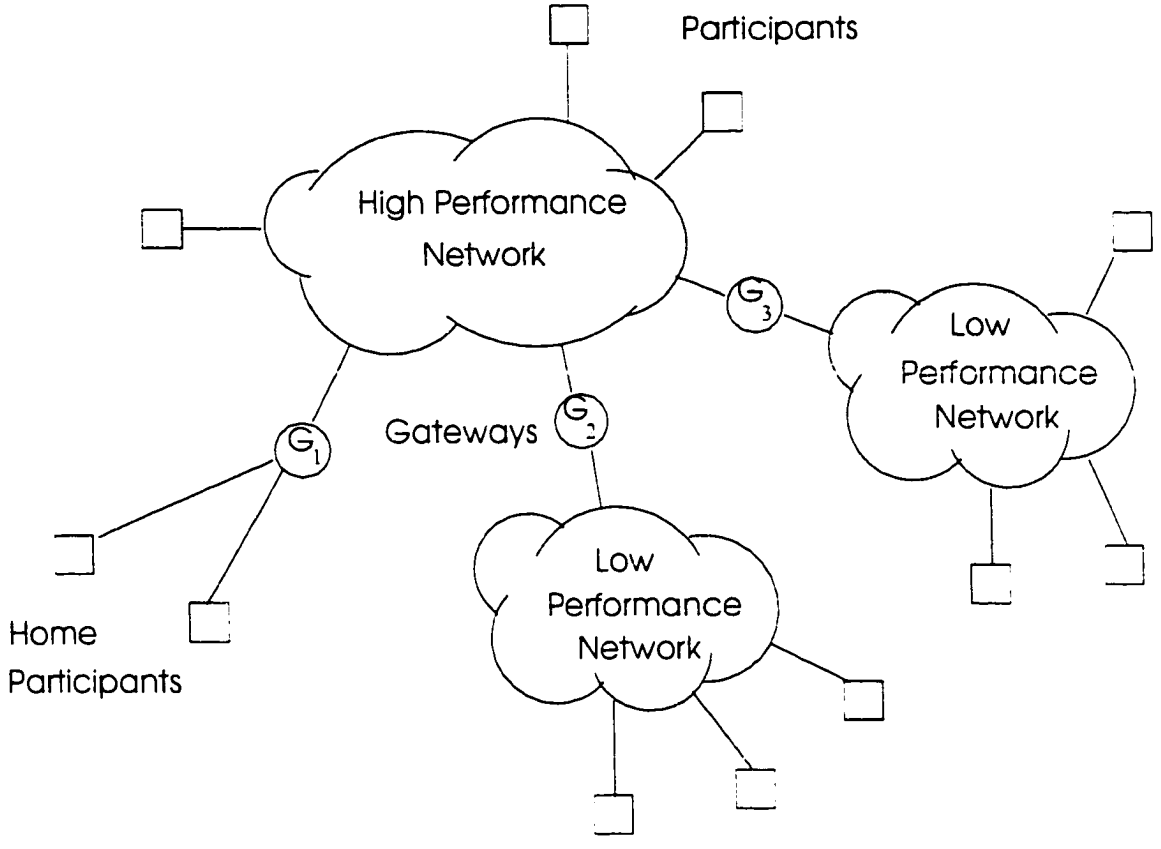


Fig. 1.1. A typical environment for multimedia applications.

is managed by a media gateway by rate limiting the media streams to match the capacities of the networks. Gateways are not only used for crossing bottleneck links, but for crossing security and addressing boundaries as well.

Figure 1.1 shows a typical environment for multimedia applications. The different participants are scattered in three different networks, as well as some connections over telephone lines. Three gateways are used to control the traffic between networks. At any time, any of the participants may be sending out a video stream, and receiving several. In such an environment, hierarchical (multi-level) encoding would be desirable.

In hierarchical encoding, each data stream is encoded as a set of sub-streams with incremental priorities. Typically, there would be a base layer data stream,

which is the most vital sub-stream and the data stream cannot be decoded without it. It contains the minimum information needed to decode the data stream at the lowest possible quality. Then, there would be one or more enhancement sub-streams. Figure 1.2 shows how an image encoded by a hierarchical technique can be decoded at several levels of detail.

Visual data streams should also have the ability to be scaled down at both the encoder and the decoder, to adapt to current network conditions and host capabilities. Going back to figure 1.1, a sender connected to the high-performance network should be able to send out a large number of enhancement sub-streams, for example one base layer and four enhancement layers. Receiver in the same network should be able to receive all streams and decode as much layers as they can process. The gateways' job is to drop some of the enhancement layers, if needed, to match the other side's capacity. For example, gateway G3 may pass through three enhancement layers while gateway G1 may pass through only one enhancement layer. The use of hierarchical encoding allows for such flexible adaptation to match network and end-host capabilities without the need for decoding and re-encoding the video stream at intermediate gateways.

1.3 Motivation

There are a lot of research and standards in the field of video encoding, and new encoding techniques are being developed regularly [2, 9, 10, 21]. Each video encoding technique provides certain desirable features and has certain drawbacks, so the question is: what will the new encoding techniques, presented in this dissertation, provide that has not been done yet?

In spite of the plethora of image and video encoding techniques available, yet their main, and sometimes their only, target is the compression ratio. In real-time applications over an unreliable network with heterogeneous receivers, there are other vital properties that must be realized in the encoding technique: scalability,

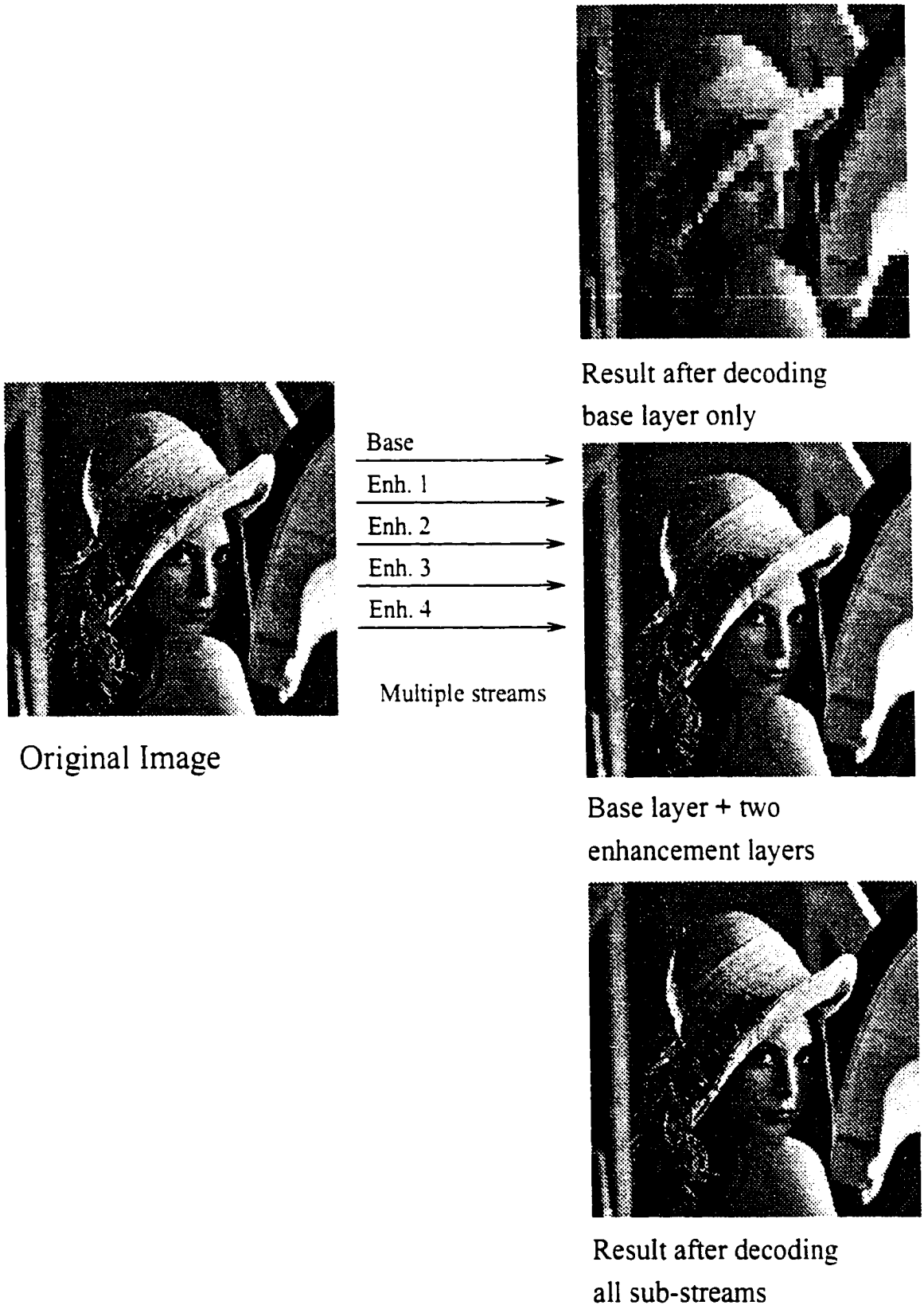


Fig. 1.2. Decoding a hierarchically encoded image.

robustness, and simplicity. Our objective is to provide an encoding technique that provides all of the properties together.

To date, all encoding techniques addressing the field of interactive image and video communication are developed based on a frequency domain transformation as a first step. The discrete cosine transform (DCT), fast Fourier transform (FFT), or wavelet transform are typically used [31]. Those transformations result in good compression ratios and may achieve scalability, but have very high computational complexities.

Another common weakness of frequency-based encoding techniques is their symmetric nature. The complexity of the reverse transformation is the same as the original transform, so decoding time is as high as the encoding time. In multi-user interactive multimedia applications, it is desirable to have a low complexity decoding technique due to the fact that each participant will, typically, be encoding only one video stream and decoding several.

Also, robustness is important when dealing with congested networks and unreliable connections. The system should be able to reproduce the frames as best as possible given only a subset of the encoded data. Robustness is rarely addressed in encoding techniques.

The fore-mentioned shortcomings of existing image and video encoding techniques, with respect to supporting interactive multimedia applications, motivated us to develop our own encoding techniques which target the specific needs of such applications.

1.4 Approach

While frequency-based techniques may be used for multi-user interactive multimedia applications, a spatial-based approach is much more simple to work with, and can yield high compression ratios as well.

Our encoding techniques are based on a hierarchical layered coding of data.

which is an efficient and flexible coding scheme. An important feature of this layered approach is to accommodate the heterogeneity between the different senders and receivers. It is not necessary for a coder to code all layers from the lowest resolution up to the required service resolution. Rather, users and service providers can choose a layer configuration that meets their application requirements.

Another advantage of layered coding schemes is their ability to add redundancy to each layer separately, thereby giving important higher layers more protection than less important lower layers. Also, layered coding schemes provide an inherent method of loss concealment for the higher resolution signals (lower layers). The low bit rate coded data from the higher layers are more protected against errors and thus, can be used to reconstruct lower quality but acceptable pictures at the upper layers in the event of packet losses.

The traditional layered encoding scheme used is the pyramid encoding technique [12, 31, 42]. In this technique the image or video frame is encoded multiple times, each time with an increasingly better quality. This provides both scalability and robustness, but at a cost of more computational time and a modest compression ratio.

Our approach is to use the quadtree and octree representations [30] as a base for developing our image and video encoding techniques. Their hierarchical structure makes them both scalable and flexible. Robustness is also achieved because a missing node in the tree can be substituted by the average of its sons' nodes or it can be approximated by its parent and brother nodes. In addition reconstructing the image or frame pixel array from the tree structure is a simple operation, especially when compared with the inverse DCT or FFT. This makes the decoding component of our encoding techniques simpler and less time consuming.

1.5 Accomplishments

We can summarize the work done in this dissertation as follows.

- We developed a new image encoding technique that satisfies the main criteria: a good compression ratio, simplicity, scalability, and robustness. The encoding technique was built using the efficient quadtree representation as a suitable data representation method and vector quantization for the actual compression.

- We used our new image encoding technique as a base for developing a new video encoding technique. We used the well-known differential approach to make use of the temporal redundancy between video frames. We obtained good, but not dramatic, results with this technique.

- Next, we used a more bold approach and applied a three-dimensional method, the octree, in encoding the video stream. The octree is typically used in encoding three-dimensional objects, and we now show that it can also be used to successfully encode video streams. We obtained extremely good results, mainly due to the flexible nature of octrees.

- Finally, we developed a hybrid technique, which uses both the octree and differential approaches to video encoding. We obtained satisfactory results with the hybrid approach, in comparison with the octree encoding technique.

1.6 Organization of the Dissertation

The rest of the dissertation is organized as follows.

Chapter 2 gives some background information about quadtrees, vector quantization and scalability. It also summarizes a survey of the different encoding techniques designed by researchers, and those available commercially.

Chapter 3 describes our image encoding technique in detail and describes the experiments we performed to prove its efficiency compared to other image encoding techniques.

In chapter 4, we describe a method of extending our image encoding technique to video. We use a well-known differential technique. We also present our experimental results in comparison to the MPEG video encoding technique.

Chapter 5 describes a more efficient way to encode video by using a three-dimensional approach. We used the octree representation instead of the quadtree. We also describe another video encoding technique which combines both the quadtree and octree techniques to obtain better performance.

Finally, Chapter 6 concludes and summarizes the dissertation and gives a list of possible future extensions to the work presented here.

CHAPTER II

BACKGROUND AND RELATED WORK

In this chapter, we give an introduction to compression techniques and we introduce some basic definitions. We describe the need for having robustness and scalability as important features in a compression system. We also describe the quadtree representation method, and the technique of vector quantization.

We then describe the different techniques developed by researchers for encoding image and video data. Finally we briefly describe the operation of some of the most well known commercial encoding techniques used. We emphasize the JPEG and MPEG techniques as those are typically used as a benchmark for the comparison of different techniques.

2.1 Basic Concepts and Techniques

All compression techniques are based on the concept of eliminating as much redundancy from the data as possible. There are two main methods: lossless compression, in which no information is lost in the encoding process and the re-constructed image is identical to the original image. The other technique is lossy compression, in which some of the information contained in the image or video data is sacrificed in order to obtain better compression results. The choice of lossless or lossy compression depends on the application in which compression is needed. A medical application is an example for a system requiring lossless compression, while videoconferencing

is an application that can tolerate lossy compression.

A video compression system should smartly combine spatial, temporal and spectral redundancy reduction techniques to obtain the best compression possible. Spatial redundancy is the redundancy between pixels in the same frame. Temporal redundancy is the redundancy between pixels in consecutive frames. Spectral redundancy is the redundancy in color between adjacent pixels. Combinations in a variety of ways specify different compression systems.

There are three combinations of the two main compression techniques: spatial then temporal compression, temporal then spatial compression, and spatio-temporal (three-dimensional) compression. They are shown in figure 2.1.

The first two methods, shown in figures 2.1(a) and 2.1(b), are simply the combination of two different coding techniques in turn so that the attractive features of both schemes could be utilized. Their main advantage is their simplicity in implementation, as each compression is done separately. However, they do not perform well in the compression aspect compared to the other technique.

In spatio-temporal coding, encoding is applied directly to the (x,y,t) space. Hence, horizontal, vertical, and temporal redundancies existing in the spatio-temporal domain can be exploited together. Various 3D coding schemes, e.g. 3D sub-band coding [26], and 3D wavelet coding [21], has been investigated for medium rate video transmission. We also develop our own spatio-temporal video encoding, described in chapter 5.

Spectral redundancy is usually reduced by converting the three red, green, and blue signals into three other signals: the luminance signal that carries information on lightness and brightness, and two color signals. As the human visual system is less sensitive to color than to luminance, the color signal may be transmitted with less accuracy. For example, the PAL standard calls the luminance Y and the two colors signals U and V. Conversion from the RGB format to the YUV format is done using these equations:

$$Y = 0.30R + 0.59G + 0.11B$$

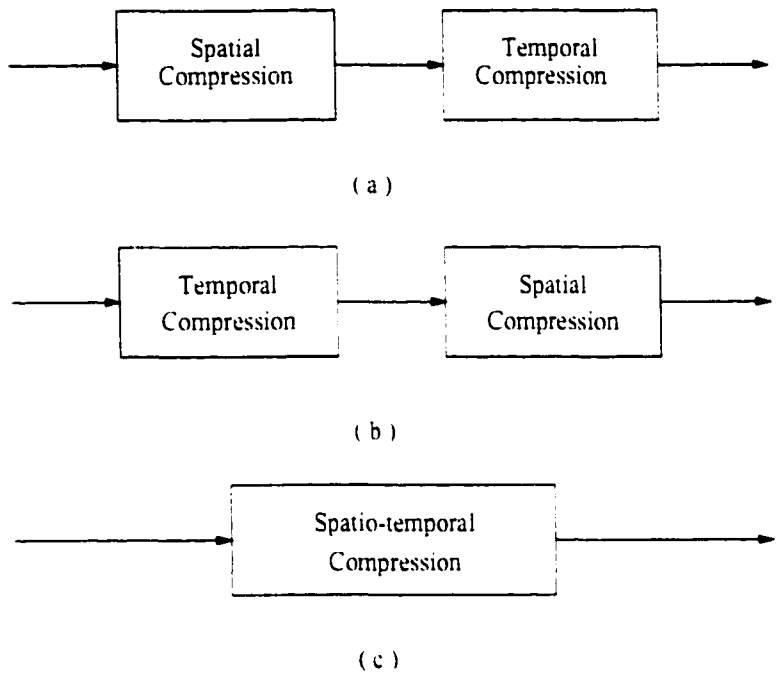


Fig. 2.1. Different video compression methods. (a) Spatial then temporal compression. (b) Temporal then spatial compression. (c) Spatio-temporal compression

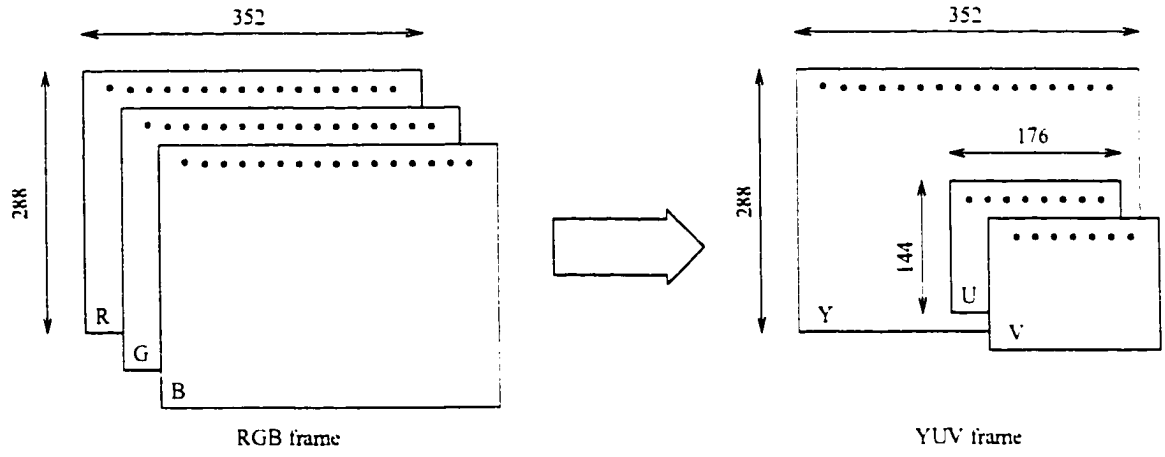


Fig. 2.2. Conversion from RGB to YUV formats .

$$U = 0.493(B-Y) = -0.15R - 0.29G + 0.44B$$

$$V = 0.877(R-Y) = 0.62R - 0.52G - 0.10B$$

The number of samples per line of the color signals is halved, and the number of lines per frame is also halved. This sampling of the color signals provides a 50% reduction in the total number of samples, as shown in figure 2.2. Also, the bit rate of each color stream is chosen to be less than that of the luminance component to achieve further reduction. The separation of the luminance component also enables compatibility with receivers that can display only gray scale images.

In this dissertation, we work only on reducing spatial and temporal redundancy, using new techniques. We experiment with s set of 8 bit gray scale images and video sequences. Our work could simply be extended to encode color images using the YUV method described above.

2.1.1 Nature of Communication

Many current and proposed telecommunication-based applications rely on the multicasting of multimedia information over packet-switched networks. In the past, protocols with packet retransmission were used. But in real-time applications, retransmission is not a practical solution because of the unpredictable additional delay.

Also, communication-based applications now usually have a large number of different capacity entities. Basically, the two main problems are:

1. Packet losses are unavoidable, irregular, and unpredictable in nature.

2. The wide variety of networks with different capacities, and end-station receivers with different processing power limits the transmission rate to the lowest capacity network, and lowest power workstation during a multicast session.

The solution lies in an encoding technique which is both robust (by adding redundancy) and scalable, to support the heterogeneous nature of multicasting.

2.1.2 Scalability

Scalability is the ability of the encoder to encode a data stream such that the data stream can be decoded at different quality levels. Thus it should be possible to extract a lower bit rate stream which will reproduce a lower resolution (or smaller) image, one having reduced frame rate, or one which can be decoded using a simplified decoder. Scalability can apply to images, video, audio, and even text. For example, an image can be divided into 3 levels, a base layer and two enhancement layers. A receiver must receive at least the base layer to be able to decode the image and quality can be improved by decoding one or both of the enhancement layers.

In digital communications applications, there are many cases where scalable (multi-level) coding is needed. One example is dealing with network traffic: instead of dropping vital information at the source or in internal network nodes, a representation that adapts itself to the network load by changing the resolution so that it complies with what the network can deliver would be an extremely useful measure against resource exhaustion. Another field is broadcasting to heterogeneous receivers, since different receivers have different processing power and capacity. The optimal solution here is to use a scalable representation so that each receiver can decode the signal according to its capabilities.

Another environment that could benefit from scalable codes is multi-point

communication with mobile hosts (or low-bandwidth connections): since mobile links are typically of much lower bandwidth than wired ones. By reducing the resolution at the base-to-mobile link, wired participants would still be able to utilize the full bandwidth available to them without being compromised by the presence of mobile participants. Also, having multiple streams of the data helps in error control. The base layers may be transmitted with more error protection and correction capabilities than enhancement layers. A typical application of this is in a distant learning system [22], where students connected from the home have much lower capabilities than those in class, so some streams, for example video, must be scaled down to accommodate them without affecting the rest of the class.

There is many research projects in the field of scalable codes. For example, at the university of Erlangen-Nuremberg [12, 13], they work on "spatio-temporal pyramids", in which the video is encoded at four different quality levels for achieving scalability. The quality variation is due to the number of bits allocated per pixel in each frame and in the number of frames. Unequal error protection is applied to the different levels.

2.1.3 Robustness

Robustness is one of the major properties required in any encoding technique designed for interactive use, since data loss during transmission is inevitable. Most encoding techniques rely on the underlying communication protocol to deal with missing data. It would be more efficient to be able to decode an image or video frame based only on partial data.

One of the projects addressing the issue of robustness is the PET (Priority Encoded Transmission) project [2, 17, 18, 20], it encodes the image or video data using a multilevel forward-error-correction scheme that provides graceful degradation of the system. The PET system is based on two main ideas: adding redundancy to each level according to its priority, using erasure codes [17], and the stripping of

data into packets. The project is very flexible as it is designed to work over previously encoded data by any multi-priority application. However, the fact that it is a separate process after the encoding means that it takes extra time and buffer space. Also, it cannot work over a scalable encoding technique, scalability is lost in order to provide more reliable transmission. Moreover, it adds an overhead of about 24 percent of the video size for error protection.

In this thesis, we achieve robustness by simply using a data structure that contains redundancy as part of its nature: the quadtree and the octree. We control the level of robustness by eliminating as much redundancy as the application requires.

2.1.4 The Quadtree Representation

In recent years, many methods have been proposed for achieving high compression ratios for compressed image storage and transmission. A very promising compression technique, in terms of compression ratios, image quality, and scalability, is the quadtree image representation [29, 30]. Another intrinsic advantage offered by the quadtree is a fast decoding time.

The quadtree is an approach to image representation based on successive subdivision of the image into quadrants. In essence, we repeatedly subdivide the image array into 4 quadrants and examine each quadrant in turn for homogeneity. This is repeated until we obtain blocks (possibly single pixels) which consists entirely of only one color. This process is represented by a tree of out-degree four (and hence the name, quadtree) in which the root node represents the entire image, the four sub-quadrants represent the four corners of the image, and the leaves (terminal nodes) correspond to those blocks of the image for which no further subdivision is necessary.

Figure 2.3 shows a sample image and its quadtree. We chose a simple black and white image for simplicity, but the same technique applies for color images. The

resulting tree is of depth three. Circular nodes (non-terminals) in the tree mean that this part of the image is of different colors and will be decomposed further. Usually, the average color of that area is stored in this node. A black node represents a black part of the image, while a white node represents a white part of the image.

Quadtrees are typically constructed by top-down or bottoms-up methods. In top-down construction [23, 30], a judgment is first made as to whether the entire block can be represented by a single leaf or whether it must be subdivided into four sub-blocks. If a block is divided, then a binary decision is made for each sub-block to determine whether it needs further division, and so on.

Bottom-up construction [4, 42, 43] consists of binary decisions to merge, where construction begins with the smallest possible block size of the quadtree. If all relevant sub-blocks have been combined into a larger block, then a decision is made whether to combine the larger blocks into a yet larger block, and so on.

Several researchers have worked on the application of quadtrees in image compression. The recursive binary nesting (RBN) technique has been developed by Clarke and Cordell [6, 7, 8, 9]. The idea has subsequently been re-investigated by Denatale et al. [10]. The whole matter of quadtree image representation and analysis has been examined in detail by Wilson et al. [48] and Todd and Wilson [45]. Sullivan and Baker [44] have developed a video encoding system based on quadtrees, vector quantization, and motion compensation. Their system has a very good compression ratio although it is time-consuming to encode, therefore it is not suitable for real-time communication.

The most obvious storage structure for a quadtree is a fully pointered tree [29] where each node record has 4 pointers to its four sons, and a 'color' field representing the node's color or the average color of its 4 sons if it is a non-terminal node. Although this method is simple and easy to implement, it needs a lot of storage space. Other quadtree representations has been proposed : Leaf-Code [49], Cube-Code [24], Gray-Code [38, 39], Tree-Code [25], DF-quadtree [16], One-to-four quadtree [41], and Goblin quadtree [47]. In this dissertation, we present new image and video

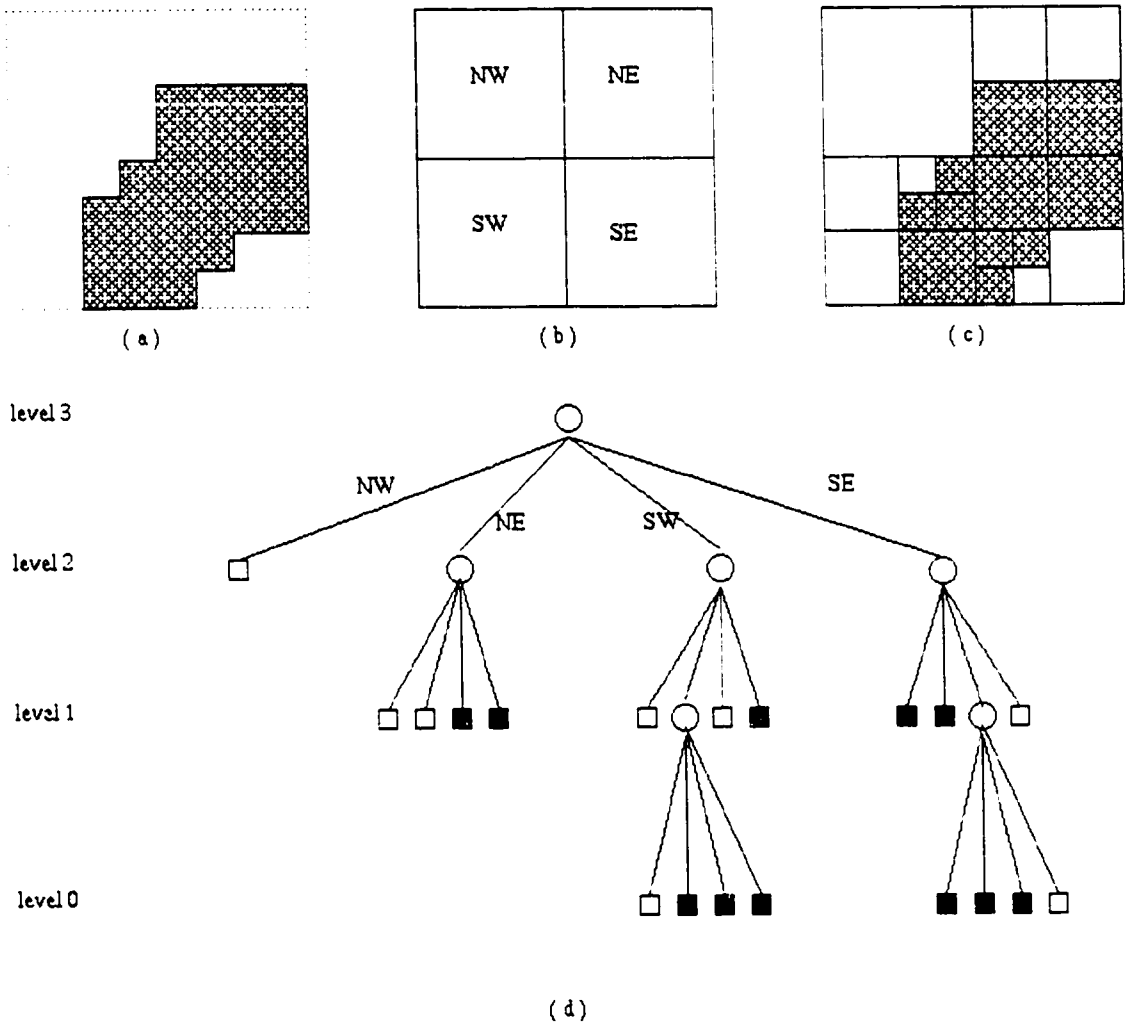


Fig. 2.3. Quadtree structure.

representation method, based on the Gray-Code.

2.1.5 Vector Quantization

Quantization is a method for coding signals such that an approximate signal is constructed from a finite set of possible values. A given sample x of the signal is specified by an index k into the finite set if it falls into the corresponding interval:

$$x_k < x < x_{k+1} \quad where \quad k \in \{1, 2, \ldots, N\}$$

where N represents the number of elements in the finite set of reconstruction values.

The integer k is transmitted to the receiver to identify the particular value in the finite set that should be used to represent the signal's amplitude at that point. The value that k maps to is called the representative level, or reconstruction value, and the amplitudes x_k represent levels or thresholds. The mapping $y = Q(x)$, the quantizer characteristic, is a staircase-shaped function. A quantizer may be classified as either *uniform* or *nonuniform*. Uniform quantizers simply divide the domain into equally spaced regions and usually use the mean of each region as the representative level; nonuniform quantizers, however, have variable representative level intervals which reflect the statistics of the data used to build the finite set of amplitudes (codebook). It should be noted that since the coded data is formed from a reduced set of values, irrecoverable information loss would occur.

Vector quantization expands the concepts employed in scalar quantization to the multidimensional case. As an example, consider a block of binary data consisting of all 1s. Transmission of the data could be accomplished by sending each element separately, achieving a bit rate of 1 bit/sample. However, if the whole block is taken together and indexed to an identical block in a codebook, only the index need be transmitted; this would yield a bit rate of $(1/xy)$ bits/sample, where x is the horizontal size and y is the vertical size of the data.

Using vector quantization, high compression ratios are possible with relatively small block sizes. Vector quantization is a mapping from an L-dimensional

Euclidean space R^L to a finite subset of R^L. This finite subset is called a vector quantization codebook or vector quantization table. By choosing the size of the codebook, the transmission rate of a vector quantization coding process can be controlled. The objective is to select an optimal codebook of size N that results in the lowest possible distortion among all possible codebooks of the same size. $\log_2 N$ bits are needed to transmit the index i if fixed length coding is employed. The rate of an N element vector quantizer may be expressed as follows:

$$Rate = \frac{\log_2 N}{L} \quad bits/sample$$

where L is the number of members in each vector. An important feature of fixed rate vector quantization is that fractional bit rates are achievable, unlike those achievable with scalar quantization.

Since the quadtree structure is by nature multi-dimensional, we can encode the colors of the four brother nodes together as a single vector and make use of the spatial dependency among them. Therefore, we treat each four brother nodes as a vector of size 4. This vector forms the input to the vector encoder. Both the encoder and the decoder have a codebook of 4-dimensional vector. The vectors in this codebook are selected to be representative of the input vectors. At the encoder, the input vector is compared to each code-vector to find the nearest match, we then transmit the index of the nearest match to the decoder. The decoding consists of a simple table look-up.

Figure 2.4 shows a simple example of vector quantization. We have a codebook of vectors, each of dimension 4. We compress a sample element (14,9,5,21). We start by finding the nearest match to it, which is element (18,10,4,18). We then represent the data by the index of the matched element, 3. The number is then transmitted to the receiver, who has an identical codebook. The receiver directly gets the vector (18,10,4,18) from the index. Clearly, it is not identical to the original data, just a best match, hence we say that vector quantization is a "lossy" compression method.

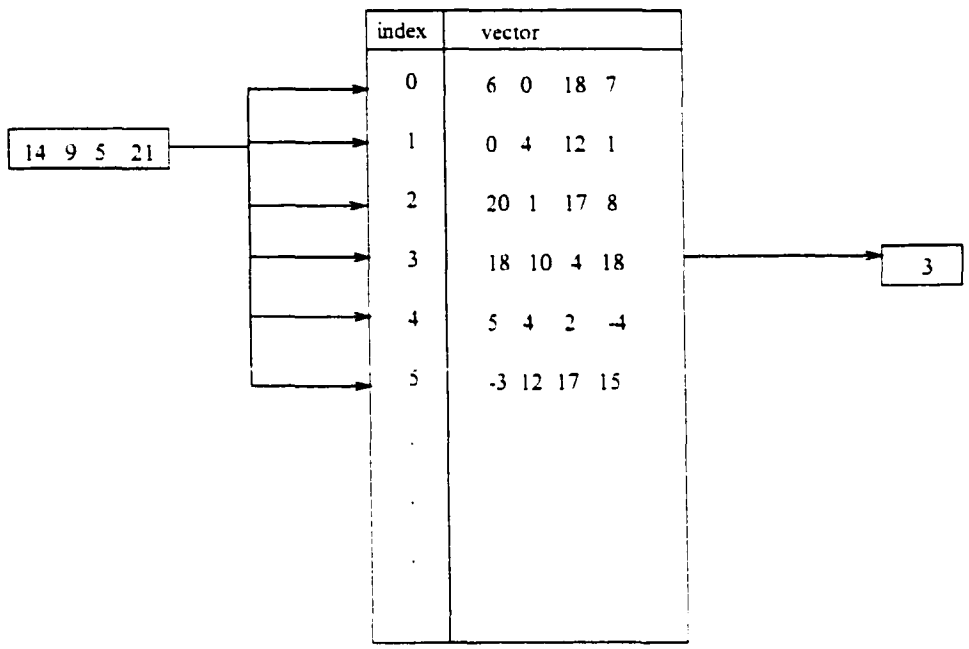


Fig. 2.4. An example of vector quantization.

Vector quantization is a very attractive encoding scheme for applications in which the resources available for decoding is considerably less than the resources available for encoding, because encoding requires a search through the code book for the nearest possible entry, while decoding is a direct table-lookup in the codebook. It is also simple and flexible enough to be included with other techniques, as we have done here by using it to encode a quadtree. The theoretical basis for vector quantization is described in [11].

2.2 Classification of Image and Video Encoding Techniques

Due to the large difference in requirements for image and video applications, there are many different methods to encode image and video data. A simple classification is shown in figure 2.5. We divide encoding methods to spatial encoding, waveform-

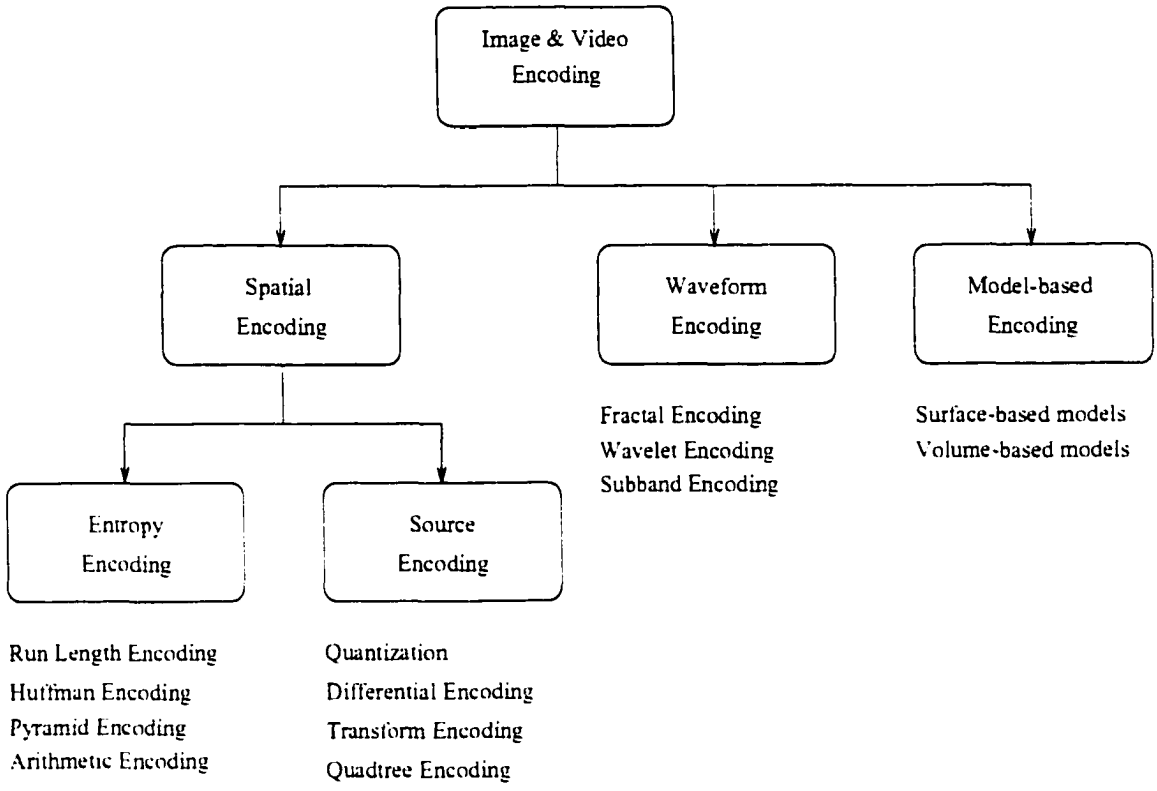


Fig. 2.5. Classification of Encoding Techniques.

based encoding, and model-based encoding. Most commercial video encoding techniques are a combination of several of these methods to obtain the advantages of each. In this section, we give a brief survey of some of techniques, which are relevant to this work.

2.2.1 Spatial Encoding

In spatial encoding, a frame is treated as an array of pixels, and compression is achieved based on exploiting the statistics of the frame and the local dependency between pixels.

Entropy Encoding

Entropy encoding is a term, which refers to the encoding, and compression techniques, which do not take into, account the nature of the information to be compressed. Entropy-based techniques treat all data as sequences of bits, without trying to optimize the compression by knowledge of the type of information being compressed. In other words, entropy encoding ignores the semantics of the information to be compressed.

A typical example is *run-length encoding* [31], in which a series of n successive characters C will be replaced by the character C itself followed by the number n. Another method is *Huffman encoding* [31] in which, for a given portion of the data stream, the frequency of occurrence of each element is calculated, and the number of bits allocated to each element depends on its frequency of occurrence.

Another very popular technique is *pyramid encoding* [12], in which each frame is encoded at several different levels of accuracy to achieve scalability and robustness. In the lowest level the original image is encoded, in the next level each four pixels are replaced by their average then the resulting smaller image is encoded, and so on. Compression can be achieved by encoding the pyramid levels in a differential method, so that each pixel is represented by the difference between its value and the corresponding value in the next higher level.

Source Encoding

In source encoding certain transformations take place which are dependent on the content of the original signal. Source encoding may produce higher compression ratios than entropy encoding, but the degree of compression is highly dependent on the data semantics.

A typical example is *quantization* (scalar or vector) in which the encoding process is replaced by a simple search for the best match among a series of patterns, and the index of the best match is then transmitted. This technique was described

in detail in section 2.1.5. Another method is the *quadtree* method, in which the image pixel array is first converted to a quadtree and then the resulting quadtree is encoded. This technique was described in detail in section 2.1.4.

2.2.2 Waveform Encoding

In waveform encoding an image is first transformed to the frequency domain and is treated as a 2D-signal waveform or a segment of an image sequence as a 3D-signal waveform exploiting the inherent statistical properties. Many image/video coding techniques, such as sub-band encoding, wavelet coding, and fractal coding, can be included in this group. The frames are first transformed to the frequency domain using a transform such as the fast Fourier transform (FFT) or the discrete cosine transform (DCT).

Sub-band Encoding

Sub-band coding [26] is based on the fact that the low spatial frequency components of a picture carry most of the information within the picture. The picture can thus be divided into its spatial frequency components and then the coefficients are quantized, describing the image band according to their importance. Lower frequencies are more important, and therefore receive more bits per component.

Wavelet Encoding

The application of the wavelet transform in data compression has been investigated by many researchers in recent years [21]. Simply speaking, the wavelet transform decomposes a signal over a set of basic functions. These functions (also called wavelet functions or simply wavelets) are generated from a unique function (called mother wavelet) by dilation and translations. Iterated operation of wavelet transform on a 2-D image signal produces a new set of data (called wavelet coefficients) in the frequency domain where the spatial structure of the image is also preserved. The

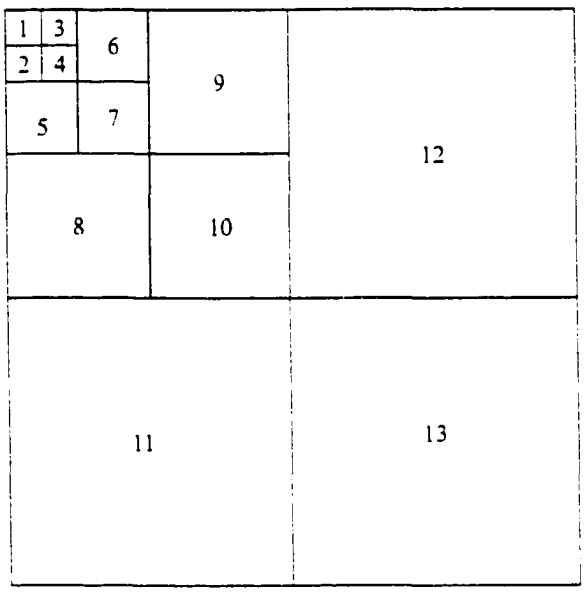


Fig. 2.6. Wavelet Encoding.

popular arrangement of these wavelet coefficients in a pyramid structure combines the wavelet decomposition and the multi-resolution signal analysis to provide a powerful tool for data compression/communication. Figure 2.6 shows how the frequency domain data is divided into different sized blocks. Since the data in the top left-side is the most important, it occupies the "top" of the pyramid by giving it more bits per pixel (blocks 1,2,3,...). Less important blocks are given less bits per pixel (blocks 11,12,13).

This pyramid-like structure is also useful in progressive transmission. Instead of sending the whole image at once, the smallest block on the top level of the pyramid is sent first for early decoding. This block is actually a coarse version of the original image and is also the lowest frequency sub-band in the frequency domain (in view of decomposed data). If desired, a higher resolution image can be obtained by sequentially sending the other blocks (higher frequency sub-bands) on the next lower level and performing an inverse wavelet transform thereafter.

2.2.3 Model-based Encoding Techniques

In model-based coding the input image or video frame is viewed as a 2D projection of a 3D real physical scene. The coding is performed by first modeling the 3D scene, and then extracting model parameters at the encoder, finally synthesizing the image at the decoder. Three key elements in model-based coding are the *modeling*, that is, the reconstruction of the 3D scene model, *image analysis* and *synthesis* based on the scene model. The image model can be classified into *surface-based* models and *volume-based* models.

In surface-based modeling, wire-frame models are most popular. In these models planar polygonal patches, e.g. triangular patches, approximate the surface. In a triangular wire-frame model, the surface shape is represented by a set of points defining the vertices of these triangles. Since the size of the patches is adjustable according to the surface complexity, wire-frame representation is flexible and general. Therefore, wire-frame models are extensively used in model based coding.

Most world objects are solids, although only their surfaces are visible. Therefore a description based on volumetric primitives is a natural approach to model objects. An often-used set of volumetric primitives is the class of generalized cylinder. In this method the object is represented by a set of primitives (cubes, spheres, and cylinders) and their relative positions. A set of deform-able cylinders were also used, this is done by using deformation parameters to control the elasticity of the main axis and the walls of a cylinder. Another popular method is to represent the object as set of constant or varying sized cubes [38].

Model-based encoding is mainly used for the restricted class where only a limited class of objects appear, e.g. the "head-and-shoulders" images used in videoconferencing. The detailed a priori knowledge about the objects can then be fully utilized to obtain high compression rates.

2.3 Survey of Common Video Encoding Techniques

In this section we review some common well known encoding techniques. We will briefly describe their method of operation and discuss the merits and drawbacks of each encoding technique.

2.3.1 Motion JPEG

Motion JPEG [46] is a simple method of encoding video by simple encoding each frame as a separate image using the JPEG encoding technique. The JPEG algorithm can be summarized as follows: The first step is to divide the image into 8x8 pixel blocks and then apply a linear transformation to each block in the input image. The linear transformation is obtained by a discrete cosine transform (DCT), zigzag ordering, and fixed-step scalar quantization. The output of the linear transformation is a 64-element vector for each block. Then, these vectors are further compressed using run-length encoding. The first step of image processing is a transformation performed by DCT. The pixel values are shifted into the range [-128, 127], with zero as the center. These data units of 8x8 shifted pixel values are defines by S_{yx}, where x and y are in the range of zero to seven. Each of these values is then transformed using Forward DCT (FDCT):

$$S_{vu} = \frac{1}{4} c_u c_v \sum_{x=0}^{7} \sum_{y=0}^{7} S_{yx} \ cos\frac{(2x+1)u\pi}{16} \ cos\frac{(2y+1)v\pi}{16}$$

where $c_u, c_v = \frac{1}{\sqrt{2}}$ for u,v=0; otherwise $c_u, c_v = 1$.

Altogether this transformation must be carried out 64 times per data unit. The result is 64 coefficients of S_{vu}.

For reconstruction of the image, the decoder uses the Inverse DCT (IDCT). The coefficients S_{vu} must be used for the calculation:

$$S_{xy} = \frac{1}{4} c_u c_v \sum_{x=0}^{7} \sum_{y=0}^{7} S_{vu} \ cos\frac{(2x+1)u\pi}{16} \ cos\frac{(2y+1)v\pi}{16}$$

where $c_u, c_v = \frac{1}{\sqrt{2}}$ for u,v=0; otherwise $c_u, c_v = 1$.

Most of the areas of a typical image consist of large regions of a single color which, after applying DCT, are represented by many coefficients with very low values. The edges, however, are transformed into coefficients, which represent high frequencies. Images of average complexity consist of many AC-coefficients with a value of almost zero. Therefore, entropy encoding is used to achieve considerable data reduction.

Merits and Drawbacks

JPEG was originally designed to encode images but has been used to encode video too due to its simplicity and robustness, and because error does not propagate from frame to frame.

As JPEG was essentially designed for the compression of still images, it makes no use of temporal redundancy between frames which is a very important element in most video compression schemes. Thus, despite the availability of real-time JPEG video compression hardware, its use will be quite limited due to its poor compression ratio in comparison with other standards designed for moving images.

2.3.2 MPEG

In the strategy adopted by MPEG [15, 19], the sequence of video frames is first partitioned into Group of Pictures (GOP) consisting of N frames. The first frame in the group (the I frame or intra-coded frame) is coded independently. The I frame is decomposed into blocks of 8x8 pixels each that are encoded without regard to other blocks and are sent directly to the block transformation process (similar to JPEG). The other N-1 frames consist of a combination of P (prediction) and B (bi-directional) frames that are coded dependently on the I frame. P-frames are frames encoded using motion compensated prediction from a past I-frame or P-frame. A prediction error is calculated between a block in the current frame and the past reference I or P frame. B-frames are frames encoded using motion compensated

predictions from past and/or future reference frames. A prediction error is calculated between a block in the current picture and the past as well as future reference pictures. Two motion vectors are calculated. The first determines the value and direction of the forward prediction (prediction calculated with a future frame as the reference). The second determines the value and direction of the backward prediction (prediction calculated with a past frame as the reference).

Scalability

Scalability was added to the MPEG standard in its second version MPEG-II [15]. The scalability tools specified by MPEG-II are designed to support applications beyond that supported by single layer video. In a scalable video coding it is assumed that, given an encoded bit stream, decoders of various complexities can decode and display appropriate reproductions of coded video. The basic scalability tools offered are: data partitioning, SNR scalability, spatial scalability and temporal scalability.

Spatial scalability involves generating two spatial resolution video layers from a single video source such that the lower layer is coded by itself to provide the basic spatial resolution and the enhancement layer employs the spatially interpolated lower layer and carries the full spatial resolution of the input video source.

SNR scalability involves the generation of two video layers of the same spatial resolution but different video qualities from a single video source. The lower layer is coded by itself to provide the basic video quality and the enhancement layer is coded to enhance the lower layer. The enhancement layer, when added back to the lower layer, regenerates a higher quality reproduction of the input video.

Temporal scalability involves the generation of two video layers whereas the lower one is encoded by itself to provide the basic temporal rate and the enhancement layer is coded with temporal prediction with respect to the lower layer. These layers, when decoded and temporally multiplexed, yield full temporal resolution of the video source.

Data partitioning scalability involves the partitioning of the video coded bit

stream into two parts. One part will carry the more critical parts of the bit stream such as headers, motion vectors and DC coefficients. The other part will carry less critical data such as the higher DCT coefficients.

Merits and Drawbacks

Due to the forward and backward temporal compression used by MPEG, a better compression and better quality can be produced. As MPEG does not limit the picture resolution, high-resolution data can still be compressed using MPEG. The out-of-order processing which occurs in both encoding and decoding side (due to the B-frames) can introduce considerable latencies. This is undesirable in video telephony and video conferencing.

Also, scalability is an added feature to the original encoding technique, which requires extra processing time to achieve it. Moreover, MPEG is very sensitive to error.

Prices for hardware MPEG encoders are quite high at the moment though this should change in the near future. Software implementation of MPEG-I decoders and encoders are already available, though not widely used for video conferencing applications.

2.3.3 H.261 and H.263

H.261 is video coding standard published by the ITU (International Telecom Union) in 1990 [14]. It was designed for data rates which are multiples of 64Kbit/s, and is sometimes called p x 64Kbit/s (p is in the range 1-30). These data rates suit ISDN lines, for which this video coder was designed. H.261 supports two resolutions, QCIF (Quarter Common Interchange format) and CIF (Common Interchange format).

The H.261 standard uses two different methods of coding: intra-frame and inter-frame. In the case of intra-frame coding, no advantage is taken from the redundancy between frames. This corresponds to the I-frame coding of MPEG.

For inter-frame coding, information from previous or subsequent frames is used: this corresponds to the P-frame encoding of MPEG. The H.261 standard does not provide any criteria for choosing which method of coding to use. The decision must be made during the coding process, depending on the specific implementation.

Similar to JPEG, for intra-frame encoding, each block of 8x8 pixels is transformed into 64 coefficients using DCT. The quantization of DC-coefficients differs from the quantization of AC-coefficients. The next step is to apply entropy encoding to the DC and AC parameters resulting in a variable length encoded word. Inter-frame coding is based on a prediction for each macro block of an image. This is determined by a comparison of macro blocks from previous images and the current image.

For H.261, quantization is a linear function and the step size is adjusted according to the amount of data in the transformation buffer. This mechanism enforces a constant data rate at the output of the coder. Therefore, the quality of the encoded video data depends on the contents of individual images, as well as on the motion within the respective video scene.

H.261 is an established standard strongly supported by the telecommunication operators. Due to the very restricted resolution in the QCIF format and reduced frame rates, the implementation of H.261 coders and decoders is possible with few, if any, technical problems.

H.261 is mostly used in applications with the dialog mode in a networked environment: video telephony and conferencing. The resulting continuous bit rate is suitable for today's wide area networks operating with ISDN and leased lines.

H.263 was designed for low bit-rate communication, early drafts specified data rates less than 64 Kbits/s, however this limitation has now been removed. It is expected that the standard will be used for a wide range of bit rates, not just low bit-rate applications. It is expected that H.263 will replace H.261 in many applications.

The coding algorithm of H.263 is similar to that used by H.261, however with

some improvements and changes to improve performance and error recovery. There are several differences between the H.261 and H.263 coding algorithms: Half pixel precision is used for motion compensation whereas H.261 used full pixel precision and a loop filter. The encoder can also be configured for a lower data rate or better error recovery. There are now four options included to improve performance: Unrestricted Motion Vectors, Syntax-based arithmetic coding, advance prediction, and forward and backward frame prediction similar to MPEG called P-B frames.

Merits and Drawbacks

The main advantage of H.261 and H.263 is their constant data rate output, which simplifies their hardware implementation. In order to achieve the constant data rate, the quality of the frames is changed, and as a consequence the encoding of image sequences with different motion parameters will not be efficient. Also, the resolution is fixed, not specified by the user.

2.3.4 CellB

CellB image compression was introduced by Sun Microsystems and is supported by its new Sun Video card. CellB is based on the techniques of block truncation and vector quantization.

CellB uses two fixed codebooks. It takes YUV images as input, the width and height of which must be divisible by 4. The video is broken into cells of 16 pixels each, arranged in 4x4 groups. A 16-bit mask and two intensities or colors represent the 16 pixels in a cell. These values specify which intensity to place at each of the pixel positions. The mask and intensities can be chosen to maintain certain statistics of the cell, or they can be chosen to reduce contouring in a manner similar to ordered dither. This method is called Block Truncation Coding. It takes advantage of the primitives already implemented in graphics accelerators to provide efficient video decoding.

Merits and Drawbacks

CellB is custom-designed for Sun Microsystems, and makes use of its graphics hardware, and therefore provides a good quality output. However, it is neither scalable nor robust and it has certain predefined resolutions only.

2.4 Summary

In this chapter, we described the basic concepts behind our new image and video encoding techniques. First, we described the nature of communication applications, and their requirements. Then, we presented the definition of scalability and robustness, two major requirements of interactive multimedia applications. Moreover, we explained the quadtree representation and vector quantization methods, which will be used in our encoding techniques.

Then, we gave a brief survey of the different image and video encoding techniques. There is basically, three different classes of encoding techniques, spatial-based, waveform (frequency-based) and model-based encoding techniques.

Finally, we illustrated some of the common commercial techniques that has been developed and is currently in use. We will later use these techniques for comparison with our work.

CHAPTER III

A NEW QUADTREE-BASED IMAGE ENCODING TECHNIQUE (QIET)

In this chapter, we present our new image encoding technique (QIET) and show its performance in comparison with similar encoding techniques. Our new encoding technique, QIET, is composed basically of three steps, first we use mean removal to prepare the image for compression. Next, we convert the image into a quadtree. Finally, we compress it using the vector quantization method.

We perform a set of experiments to test its performance against other techniques in the literature. We compare its compression results, scalability feature and robustness against other techniques. Our technique had reasonable results in compression, and superior results in both scalability and robustness.

3.1 Functional description of QIET

Our new encoding technique [33, 36] can be summarized as follows: First we apply the mean removal technique to the image to obtain a smoother image to be encoded. The mean image is encoded separately and used as a basis for reconstructing the full image. Then, for each block of the mean-removed image we obtain the quadtree structure. Finally, the colors of the nodes themselves are encoded using vector quantization. Decoding an image is the reverse of the encoding process: first we decode the vector quantized quadtree data using the codebooks, then we convert

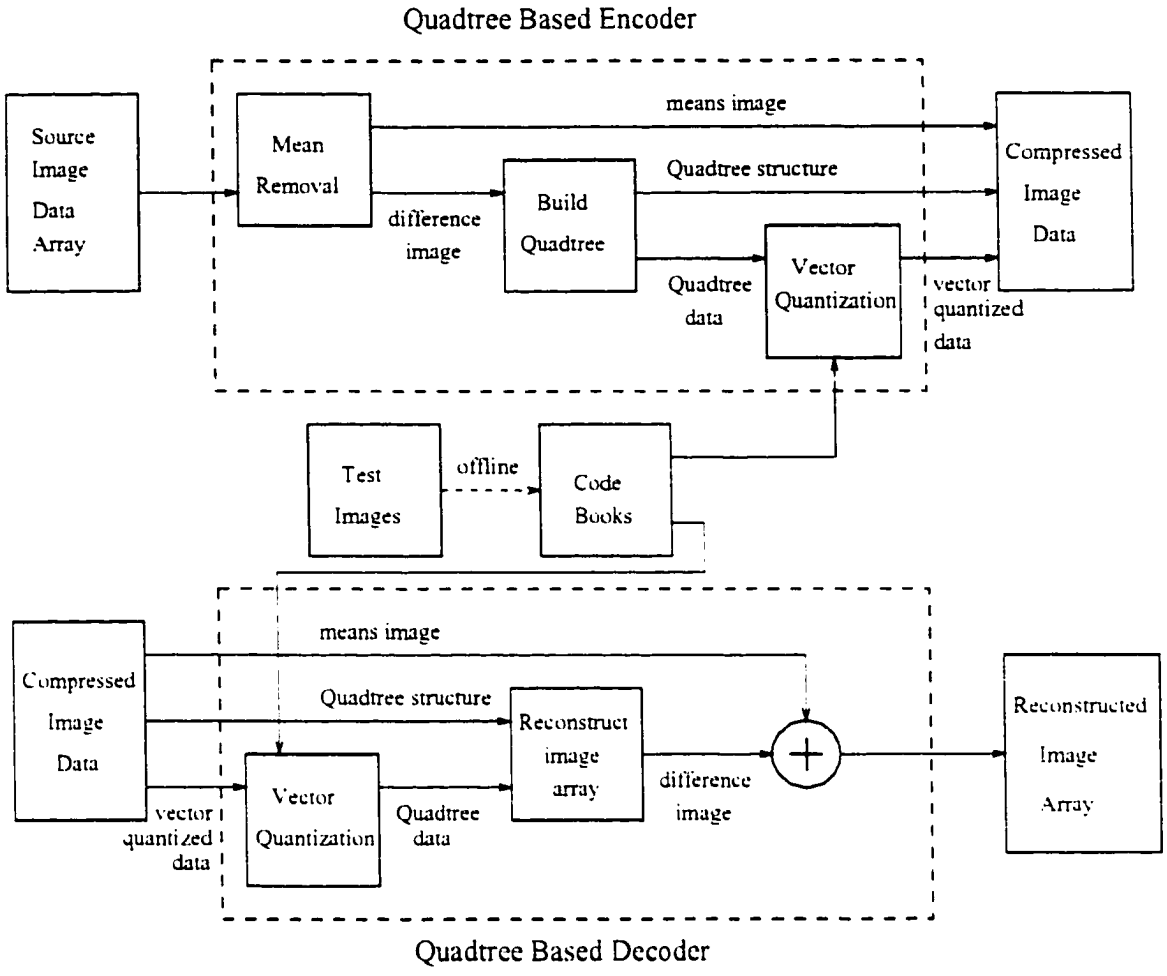


Fig. 3.1. Block diagram of the new encoding technique.

the quadtree structure into array structure. Finally we add up the means image to obtain the reconstructed image. The vector quantization codebooks are built off-line using a set of training images. Figure 3.1 shows a block diagram of the encoding and decoding techniques. In the following sections, we explain each of these steps in detail.

3.1.1 Mean Removal

Mean removal is the first step of our image encoding technique. It is used for more efficient use of vector quantization [31]. The reason is that while generating a

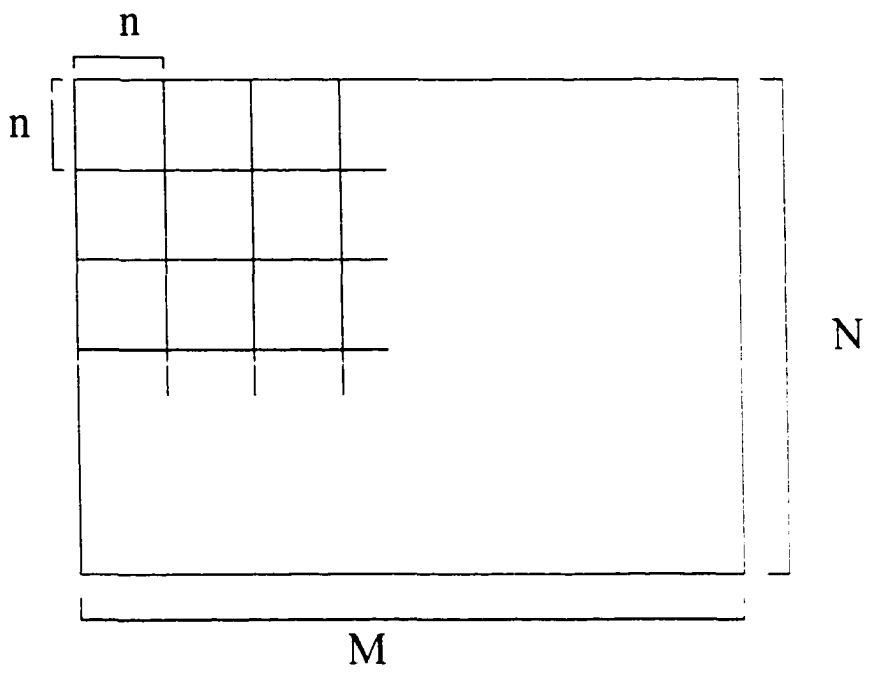


Fig. 3.2. Division of an image into square blocks.

codebook from an image. different amounts of background illumination would result in vastly different codebooks. This effect is reduced by removing the means before quantizing the image.

We first calculate the means of each (nxn) block. and then subtract the value of the mean from each pixel in that block. The mean and the mean-removed (difference) images are then encoded separately. The main issue here is how to choose the block size (nxn) relative to the image size (MxN). Figure 3.2 shows how an image of dimensions(MxN) is divided into an array of squares of dimension (nxn).

If the block size is relatively small, then the compression will not be efficient. as the mean image size will be too large. and the block identifier will need more bits and the resulting quadtree will be too shallow to provide useful scalability.

On the other hand. if the block size is relatively large compared to the image size. then a larger quadtree will be needed. and consequently more errors will be unrecoverable.

In most techniques, the block size is fixed (for example 8x8) for all images.

We believe that, for best results, the block size (nxn) should be dependent on the image size (MxN), so large images could have large blocks and small images should have smaller blocks. The minimum and maximum block sizes are calculated by the following equations:

$$n_{MIN} = 2$$

$$log_2(n_{MAX}) = \lceil log_2(Max(M, N)) \rceil$$

We choose the block size n to be a compromise between the two values, so that the log of the block size is equal to half of the log of the maximum block size. We use this equation to calculate n:

$$log_2(n) = \left\lceil \frac{log_2(Max(M, N))}{2} \right\rceil$$

So for a (512x400) image, the minimum block size is (2x2), the maximum block size is (512x512) which is just one block. By using the above equation, we choose the block size to be 32x32.

Since the image dimensions, M and N, do not have to be factors of 2, we complete the border blocks with all white pixels. To illustrate, a (512x400) will be divided into an array of (32x32) square blocks, arranged in 16 columns and 13 rows. The last row of blocks will contain 16 rows of all white pixels. Therefore, we are actually encoding an image of size (512x416) pixels.

Since the mean image is typically small, we do not quantize it. The overhead of sending the uncompressed image is very small. For example, for a 512x400 image whose block size will be 32x32 according to the previous equation, the mean image size is 16x13 which needs 1664 bits to encode (uncompressed). Naturally, an image of that size does not need compression.

Figure 3.3 shows a (512x512) image together with its difference image. The mean image is a small (16x16) image array, and the difference image is the same

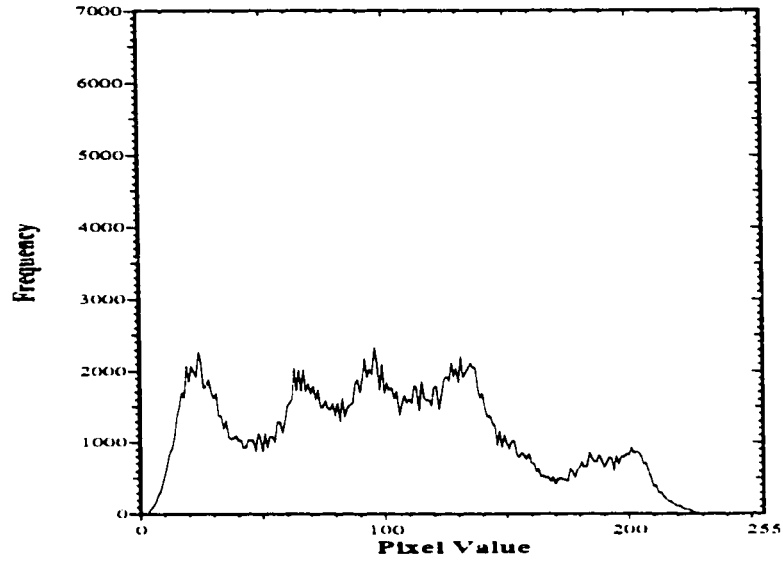
Fig. 3.3. The Lena image and its difference image.

size as the original image (512x512). The advantage of mean removal is also demonstrated in figure 3.4. We have an 8-bit gray scale image, and the pixel gray scale color was distributed among the 256 different values, as shown in the graph of figure 3.4(a). After mean removal, the distribution becomes greatly localized in the middle region, although the values have twice the initial size, as shown in figure 3.4(b). This locality will help in obtaining a better compression when we apply vector quantization to the data.

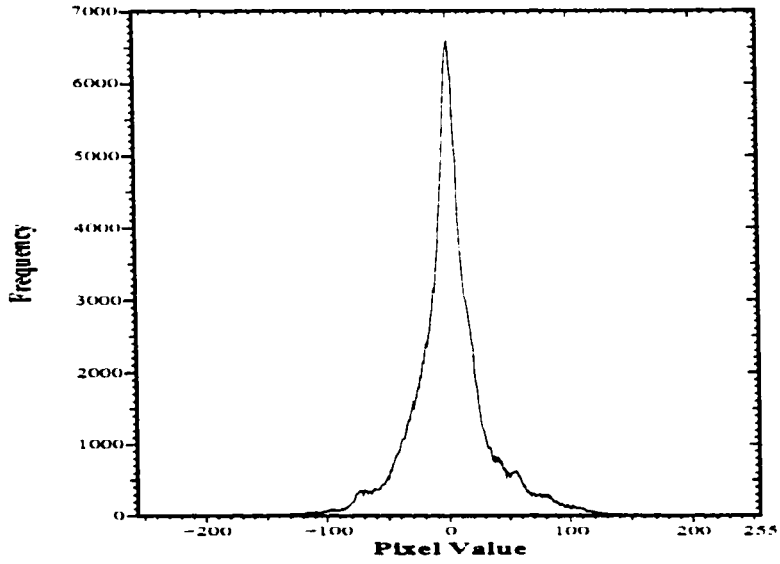
3.1.2 The Quadtree Construction Algorithm

In this section we explain how the quadtree structure is built from the difference image. The algorithm is a modification of the algorithm used in [30]. The algorithm generates both the binary quadtree structure and the node representations.

The quadtree algorithm examines each pixel in the difference image array only once in a manner analogous to a postorder tree traversal. The main procedure in figure 3.5 is termed BUILD and is invoked with two parameters, the value of the



(a) Before mean removal



(b) After mean removal

Fig. 3.4. Effect of mean removal on pixel color distribution of the Lena image.

```
procedure BUILD(IMAGE, LEVEL)
/* construct a quadtree encoding for the 2^LEVEL x 2^LEVEL image in pixel
array IMAGE. The origin is assumed to be the SE-most pixel of the image */
begin
   Remove the means from the image and encode separately.
   preload global integer array XF['NW','NE','SW','SE'] with 1,0,1,0 :
   preload global integer array YF['NW','NE','SW','SE'] with 1,1,0,0 :
   preload global character array CD[0,1,2,3] with '0','1','2','3' :
   set CODE = NULL string :
   result = CONSTRUCT(LEVEL,2^LEVEL, 2^LEVEL, CODE):
   return:
end
```

Fig. 3.5. Encoding Algorithm.

log of the image length (LEVEL for a $2^{LEVEL}x2^{LEVEL}$ image) and the name of the image array. For example, if our block size is 32, then $LEVEL = 5$. This algorithm initializes the global data and initiates the construction of the quadtree. If an image is all of one color, it creates the appropriate one-node tree.

We have four global values, integer array XF and YF that provide information about the current location of the quadrant in processing. The default is SE (0,0). To move up to NE, we add 1 to the YF component to be (0,1). To move left to SW, we add 1 to XF component to be (1,0). To move to NW, we add 1 to XF and 1 to YF to be (1,1). The array CD is used to hold the symbols for the four quadrants. For clarity, we use the number 0 to 3. CODE is the variable to hold the locational code in use. For example, if CODE = "21", we are in the NE quadrant of the SW quadrant of the main block. It also denotes the size of the current quadrant. If the block size is 32x32, the quadrant with a CODE of length 2 is of size (8x8).

The actual construction of the tree is performed by procedure CONSTRUCT, shown in figure 3.6. The procedure starts by recursively calling itself four times to get the average color of the current four quadrants. If we are at the lowest level, a leaf, we just return its color.

Next, we calculate the average of the four values and get the maximum absolute deviation from it, called DEV. For example, if the four colors are (13,3,19,29), the average is 16, and DEV is 13. Then we make the decision of whether to keep it as a gray node, or reduce it to a leaf node. If all four sons are leaf nodes, and the DEV is less than a cut-off threshold value, (typically 20-40), then we consider that node as a leaf with color equal to the average of its sons. This threshold is used to control the encoding scalability, because as we increase the threshold, more nodes are converted to leaf nodes. The threshold is a parameter of the system, and its value affects the performance of the encoding technique. The threshold used is variable and depends on the level of the node. Typically, nodes at high levels contains more data than those at lower levels, so we give them a higher threshold [44].

If we decide to keep the node as a gray node, then we compress it and send it out to the underlying communication protocol to send it to the receivers. We compress the four color field collectively as a vector using our vector quantization codebooks. The data that is transmitted is the codebook index, quadtree structure, and the locational code for that block.

Since our algorithm is based on the basic quadtree-building algorithm [30], it has a low complexity. The execution time of this algorithm is proportional to four-thirds the number of pixels in the image since it is the number of times procedure CONSTRUCT is invoked (and equal to the number of nodes in a complete quadtree for a 2^{LEVEL}x2^{LEVEL} image). The algorithm is recursive and the maximum depth of recursion is equal to the log of the image size (for example, 5 for a 32x32 block).

```
integer procedure CONSTRUCT (LEVEL, X, Y, CODE):
/* Generate the code corresponding to the 2^LEVEL x 2^LEVEL
portion of the pixel array IMAGE whose SE-most pixel is IMAGE[X,Y] */
begin
  integer array RES[4] : integer AVERAGE, DEV:
  if (LEVEL == 0)
    then return (IMAGE[X,Y]) /* At a pixel */
    else
      begin
        LEVEL = LEVEL - 1: DEV =2^LEVEL :
        for i = 0 to 3 do
          RES[i] = CONSTRUCT(LEVEL,X-XF[i]*D,Y-YF[i]*D,CODE+CD[i]):
        AVERAGE = average(RES[0],RES[1],RES[2],RES[3]):
        DEV = Maximum absolute deviation from the average :
        if (all sons are leaf nodes) and (DEV < certain Threshold)
          then
            return (AVERAGE)
          else
            begin
              Search the codebook to find the nearest match to the
              colors of the four color fields.
              Create a structure for the current node which includes
              the Codebook index and the locational code.
              Store the types of the four sons in the quadtree
              structure list (0 for leaf, 1 for non-terminal node).
              return (AVERAGE)
            end
      end
end
```

Fig. 3.6. Quadtree construction algorithm.

3.1.3 Image Representation

The next step after mean removal is to convert the difference image into a quadtree. We developed a new quadtree representation that makes it particularly suitable for compression using vector quantization. In previous work, we defined the "Gray Code" image representation method [39]. A brief description of this representation follows.

The Gray-Code method stores the quadtree as a group of several lists, each list representing one level of the quadtree. In each list, we store data for the non-terminal (Gray) nodes only, and hence the name gray-code . Each node is represented by five fields. For each node, we store its locational code [49], and four fields containing the average color of each of its four sons. The locational code specifies both the size and the exact location of the quadrant in the image.

Our new quadtree representation is based on the Gray-Code with some modifications to obtain better compression ratios. Since we used mean removal of (nxn) blocks, the resulting forest of quadtrees will each be $log_2(n)$ levels deep. Each quadtree represents an area of size (nxn) of the image. Each tree is represented by its locational code and a series of bits that shows its quadtree structure. Since all trees represent the same sub-image size, the locational code does not indicate the image size, it only represents the location. Since the blocks are in array format, the locational code is simply the concatenation of its array indices, as shown in figure 3.7.

The bits needed to encode the locational code are dependent on the image size. For a (MxN) image divided into (nxn) blocks we need (XxY) blocks where

$$X = \left\lceil \frac{M}{n} \right\rceil$$

and

$$Y = \left\lceil \frac{N}{n} \right\rceil$$

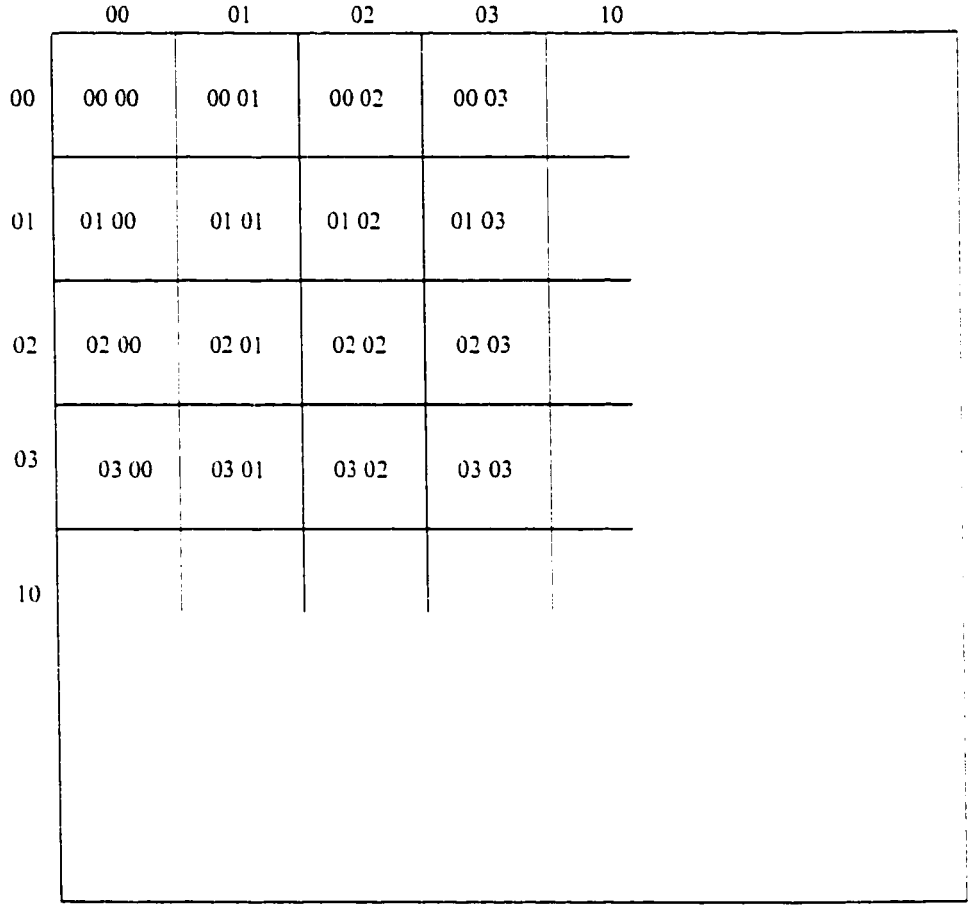


Fig. 3.7. The Locational code for our array of quadtrees.

Level	Structure
5	1
4	1011
3	1000 0011 0011
2	1000 1110 0010 1001 0101
1	0000 0000 0000 0000 0000 0000 0100 0000 0000
0	0000

Fig. 3.8. Quadtree structure for a sample block of the Lena image.

Therefore the number of bits needed to encode the locational code is:

$$N_{Loc.Code} = \lceil log_2(X) \rceil + \lceil log_2(Y) \rceil bits$$

Next, We send the structure of each quadtree in turn. We indicate termination by a leaf with a '0' and branching into child nodes with a '1'. For example the quadtree of figure 2.3 is represented by 1 0111 0000 0100 0010 0000 0000. Since the lowest level consists of all leaves we do not need to represent it. Figure 3.8 shows the quadtree structures for a single 32x32 blocks of the Lena image.

From the figure, it is easy to deduce what nodes will be sent for that quadtree. Level 5 is always 0 (block is all of the same color) or 1 (Quadtree structure following). For each other level, each element represent the structure of a node that will be received. The position of the node depends on its order and information from the higher level. For example in figure 3.8, when we receive the first item in level 3 (1000), then we know the blocks exact location: the NW quadrant of the block (from its order), its size 8x8 pixels, and that it has one non-terminal son and three leaf sons. So we expect in the next lower level to receive the structure of that single non-terminal son.

Clearly, the quadtree structure is of very high importance, since without it we cannot put an incoming node in its correct location. The structure can be

communicated separately, using reliable communication, or it can be included with the node information, which is much less reliable and can cause errors in any level to propagate to the next level. Therefore, we choose to send the quadtrees structure first separately, before sending the rest of the information.

3.1.4 Vector Quantization

Vector Quantization is used to obtain an efficient bit allocation scheme for the color data stored in each node. We want to minimize the distortion of quantizing the nodes at each level. We apply vector quantization to each level of the quadtrees separately, since the contribution of each node to the image area is proportional to the size of the node, which is determined through its level. Therefore, more accuracy is needed for nodes at higher levels compared to nodes at lower levels.

Each quadtree node is represented by the locational code of the quadtree, the quadtree structure for that block, and the colors of its four sons. The locational code and the quadtree structure were explained in the previous section. The colors of the four sons represent the average colors of the four quadrants of this node. Since this part represents the largest part of the data, we compress it using vector quantization.

The four color fields would require 9 bits each to encode losslessly. This is because each color is actually the difference between the real color of the area and the mean image so its range is from (-256) to (+255), however there is a high level of locality in the pixel distribution of any image, as shown by figure 3.4(b). Most pixel colors are actually close to the mean color. Therefore we can quantize the colors to use less bits with a low level of distortion, one pixel at a time (called scalar quantization). We carry this idea one point further by using vector quantization to quantize all four color fields together, so instead of using 36 bits to encode them, we make use of the spatial locality between them and use less bits to encode them. The number of bits chosen is a parameter of the system. The smaller the number

of bits chosen, the more compression we achieve, but at a cost of less quality.

Vector quantization is based upon the use of a codebook. Since all levels of the quadtree are not of the same importance, we use a different codebook, with a different size, for each layer. The codebook size reflects the level of detail of a layer, the higher a layer, the larger its codebook becomes. We use a codebook of size 16 items for the lowest layer (4-bit index) and double the size for each higher layer. This hierarchy of codebooks also saves decoding time, because naturally there are more nodes in lower layers, which need to be decoded faster.

The codebooks are generated off-line as follows. We use a set of training images, shown in figure 3.9. We build the quadtrees of the difference images of each picture. Then we build the codebook for layer 1 using all nodes in all quadtrees in layer 1, then layer 2 and so on. The same technique is used for all layers, the only difference is in the codebook size. We use the modified LBG algorithm [32] for codebook design, which is basically a clustering algorithm: given a set of M input 4-dimensional vectors, we find the m centroids that result in the smallest possible mean square error.

Having built the codebooks, we reference them in both the encoding and the decoding process. At the encoder, we replace the four color fields by the index to the nearest match in the codebook of the corresponding level. The distance criteria we use are the minimum mean square error. In case of equality of distance, we simply choose the first match found. To speed up the search for a best match, we can make it stop when it finds a match with a mean square error less than a certain small threshold.

At the decoder, we replace the index by the matching vector from the codebook to obtain an approximation of the original four colors. The number of bits used to encode each node is dependent on the image size (through the locational code field size, which is constant for a single image) and the level of the node (through the codebook index).

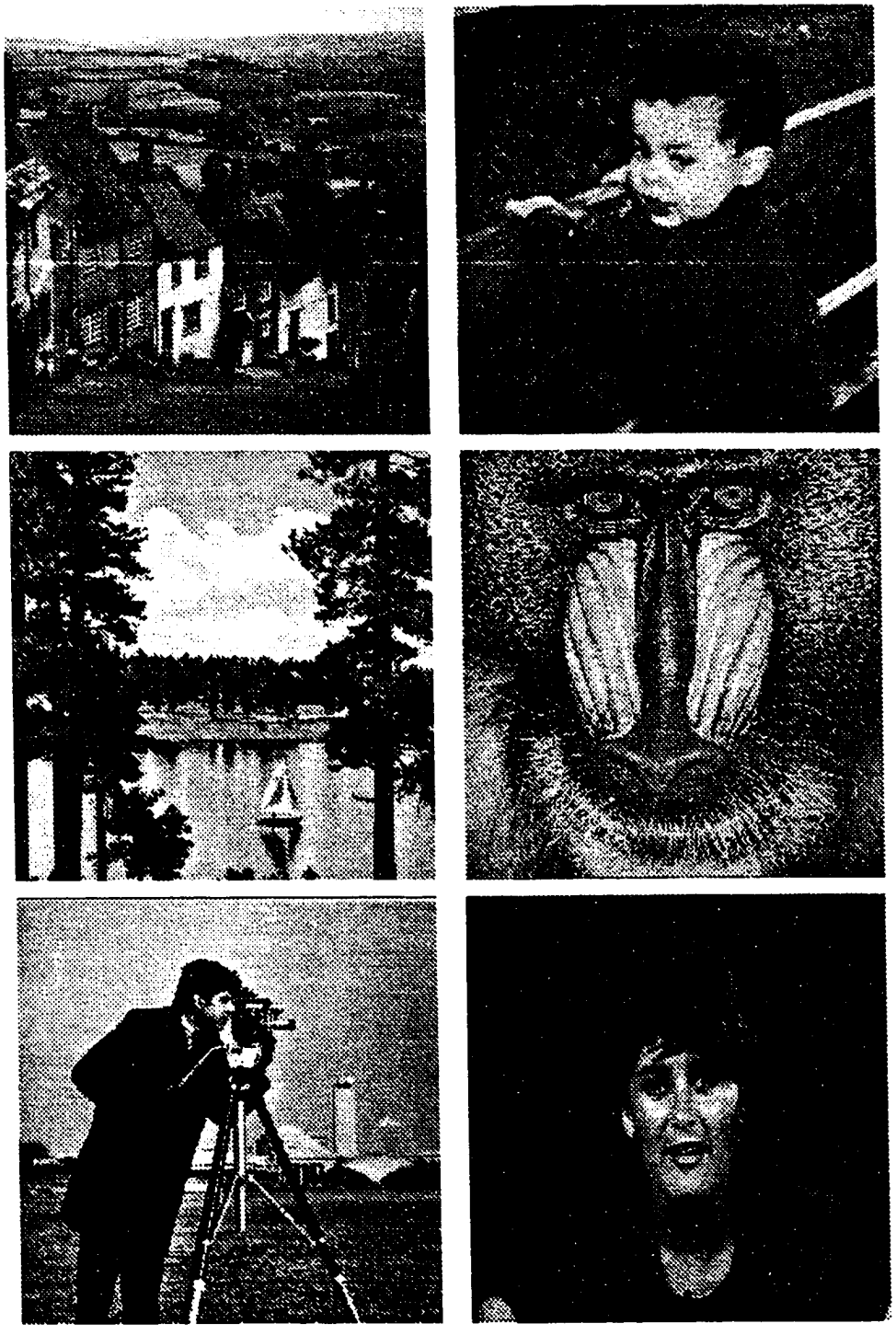


Fig. 3.9. Images used in training the Codebooks.

3.1.5 Decoding Algorithm

The decoding algorithm is the reverse of the encoding algorithm. However, it is much more time-efficient. The decoding steps are:

- Start by generating our basic image by enlarging the mean image to the actual image size. For example, if our block size is (32x32), then each pixel in the mean image would be enlarged to cover a square (32x32) area. Figure 3.11(b) shows the base image of Lena after displaying the mean image only.

- Perform reverse vector quantization to obtain back the quadtree node color data. The receiver has the same codebooks as the sender, and uses them to find the data vectors from the codebook indices. This is a direct lookup in the codebooks, not a search like in the encoding process.

- Given the quadtree node color data, and the quadtree structure, re-build the quadtree structure. Only as many levels as the receiver can process should be decoded. Since the levels are processed in a decreasing order, we can build the quadtrees of each block as we process the nodes. Each level gives information about the following level's data structure.

  Figure 3.10 shows how the quadtree structure is formed for our sample block of the LENA image. Figure 3.10(a) shows the structure after receiving the level 5 quadtree structure. It is a single nonterminal node, so we expect one node to arrive in the next level (number of nodes in following level = number of 1s in this level's quadtree structure). Figure 3.10(b) shows the quadtree after receiving level 4 (1011), now we expect 3 nodes in the next level. And so on, until we reach a level with all 0s meaning that there are no more non-terminal nodes to come.

- As the quadtree is being decoded, we plot the data obtained directly on the basic image to enhance it, in a layer by layer approach. Figure 3.11 shows

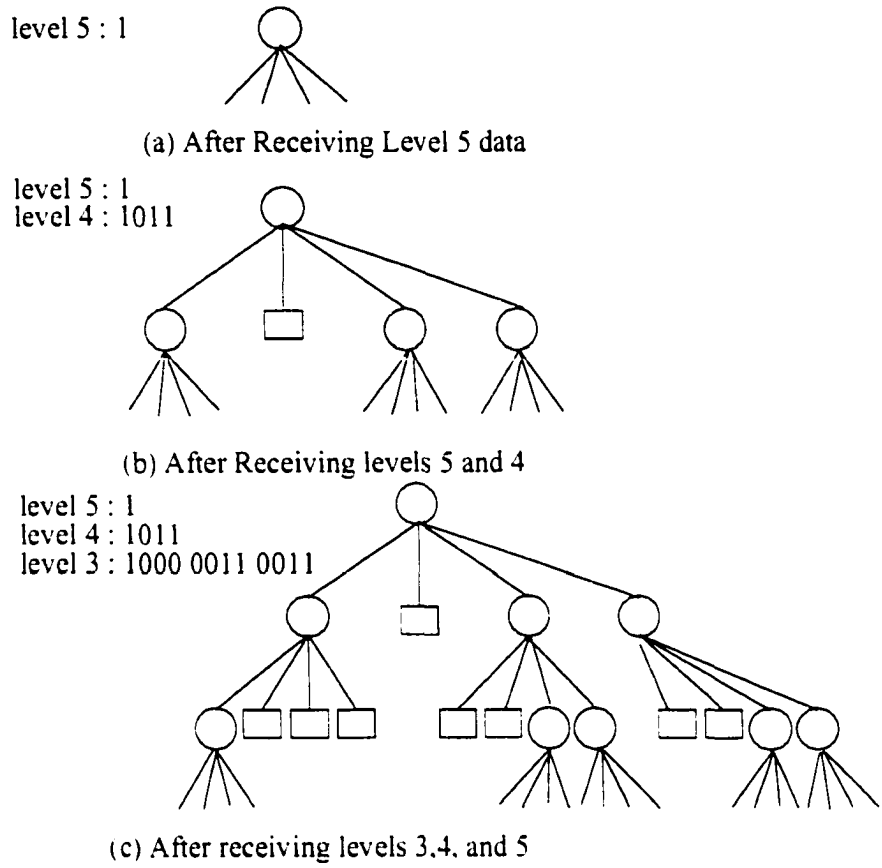


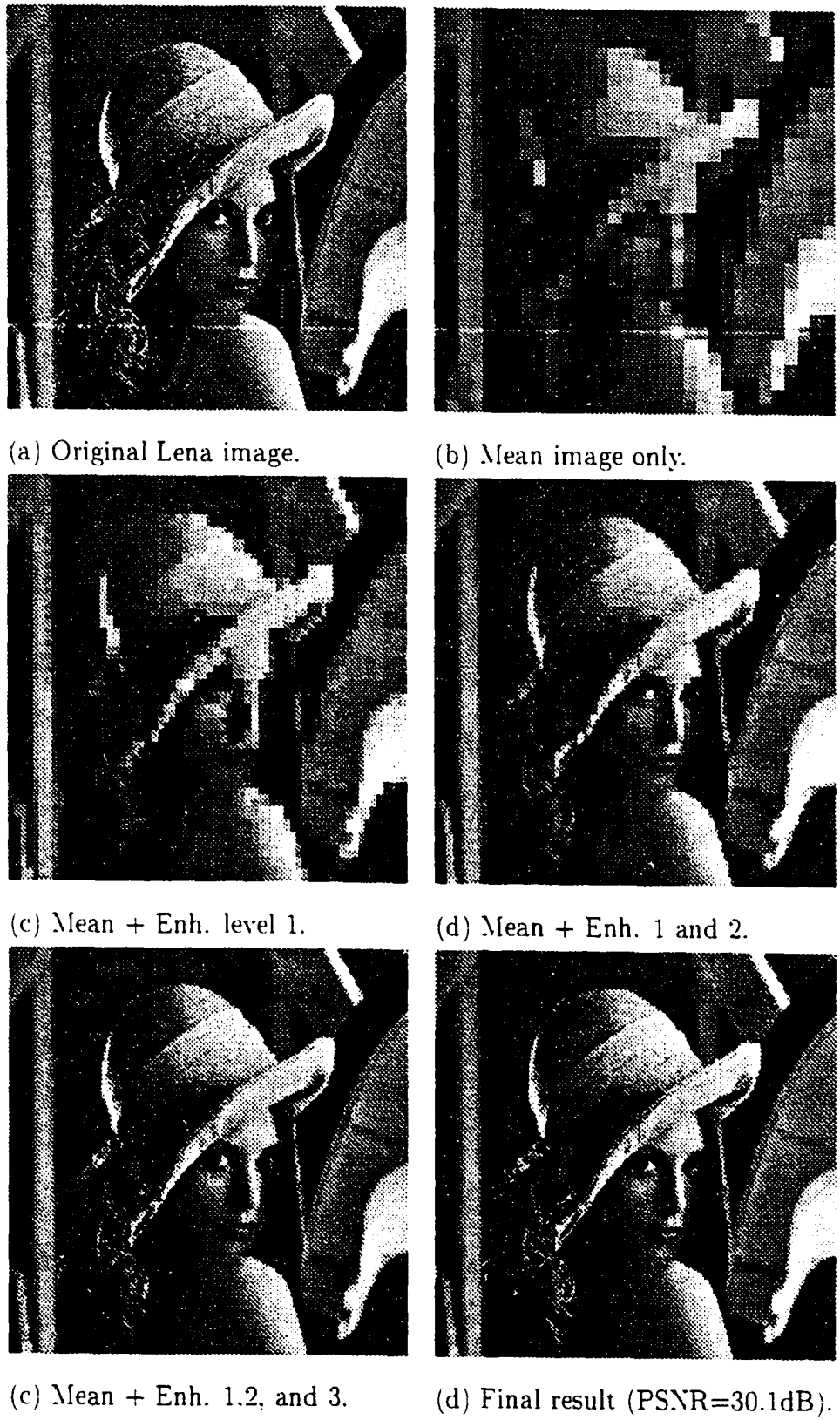
Fig. 3.10. Progressively building the quadtree structure of a sample block of LENA

the progressive building of the reconstructed LENA image. This algorithm is explained in detail in the following subsection.

Quadtree Destruction Algorithm

The decoding algorithm is simpler than the encoding algorithm. It is shown in figure 3.12. The input to the decoder is the following: the codebook for each layer, the means image, the quadtree structure, and the node color information for each level. The codebooks are fixed and are used for all images, so they do not need to be transmitted from the sender with each image.

We first load the mean image array and the quadtree structure and then process each level in turn starting from the highest level. We do not have to process



(a) Original Lena image. (b) Mean image only.

(c) Mean + Enh. level 1. (d) Mean + Enh. 1 and 2.

(c) Mean + Enh. 1,2, and 3. (d) Final result (PSNR=30.1dB).

Fig. 3.11. Progressively building the LENA image, layer by layer.

```
Read means image:
Read Quadtree structure:
Load required codebooks:
set QTREE pointer = null:
Display Means image, enlarged to the real image size.
For each required level do
  begin
    Read node data, INDEX.
    Get the 4 quadrants color using the INDEX and the codebooks.
    Get the Actual location of the four quadrants from the QTREE pointer:
    Add the mean value of this block to all four colors to get
    the actual quadrant color.
    Plot the quadrants in their positions.
    Move the QTREE pointer to the next expected node to arrive.
  end
```

Fig. 3.12. Decoding algorithm.

all levels, only as much as the receiver requires. We process the data in each level as it arrives. We already know the size it represents (from the level number) and what part of the quadtree it is (from the high-priority quadtree structure data received). We decompress the data using the codebooks, and then plot the color areas over the means image.

The execution time of this algorithm is proportional to one-thirds the number of pixels in the image since it is the number of node data received (and equal to the number of gray nodes in a complete quadtree for a 2^{LEVEL}x2^{LEVEL} image).

3.2 Analysis and Results

In this section, we show the results of our new encoding technique and compare it to other algorithms based on Quadtrees [40], and other well-known algorithms. We compare them according to their ability to compress an image, scalability, and robustness.

3.2.1 Distortion Measure

Before presenting our experimental results, let us define the distortion measure. Different researchers have used different measures, the mean square error (mse), the signal to noise ratio (SNR), or the peak signal to noise ratio (PSNR). For comparison purpose to other reported work, we use the measure peak signal-to-noise ratio defined as follows: let x(i,j) be a pixel at the i,j coordinates of the source image and let y(i,j) be a pixel at the i,j coordinates of the reconstructed image. The image size is MxN and uses 8 bits/pixel.

$$PSNR = 10 \log \left\{ \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \frac{256^2}{[x(i,j) - y(i,j)]^2} \right\} dB$$

We get the square error by subtracting the value of the pixel at the original and reconstructed image, and then squaring it. Since we are using 8 bits/pixel images, our pixel value can range from 0 to 255, so our peak signal value is 256. The PSNR reflects the quality of the encoded image, the higher the PSNR, the better the quality of the encoding technique. A typical value for PSNR is between 20 and 40 dB.

3.2.2 Image Complexity

The presented image representation technique was tested on three standard 8 bit gray-scale images used for testing encoding techniques. The size of the source pictures are 512x512 and they are shown in figure 3.13: Lena, airplane, and peppers.

(a) Lena

(b) Airplane

(c) Peppers

Fig. 3.13. Test Images.

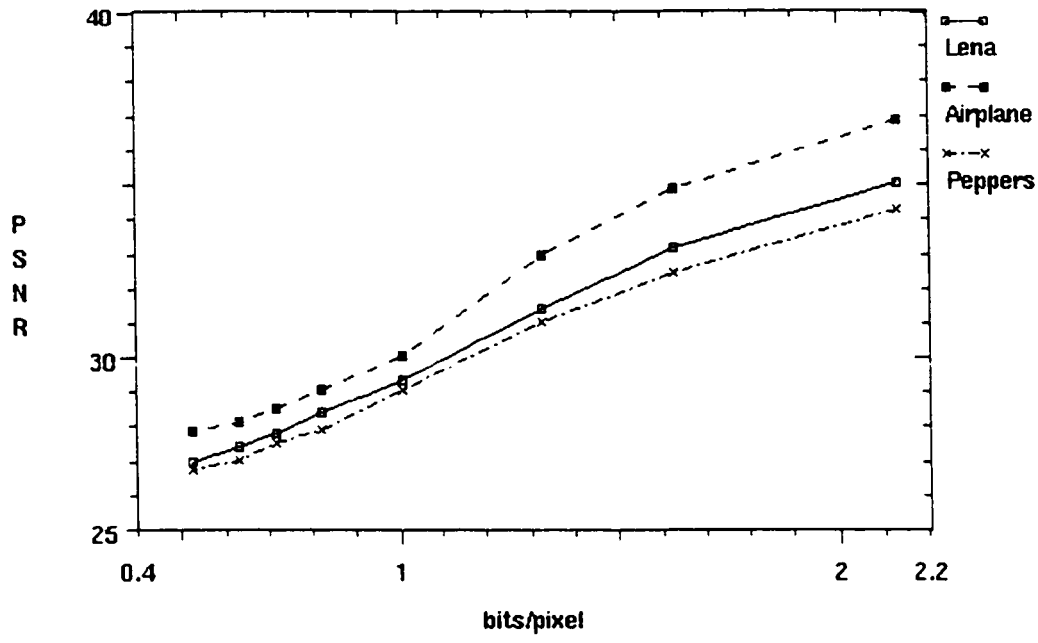


Fig. 3.14. RD Function for the Test Images.

In the graph of figure 3.14, we plot the R-D (Rate vs. Distortion) functions for the three test images. We plot the bits needed per pixel against the PSNR, which represents the quality, of the resulting image. The bits needed per pixel is calculated by adding up the total data sent for the image, including the quadtree structures and the means image, and dividing it by the number of pixels in the image. We vary the compression rate of our encoder by changing the cutoff threshold used in building the quadtrees.

For example, encoding the peppers image at 1 bit/pixel (a 1:8 compression ratio), gave an image of quality 28.7 dB. The airplane figure gives a higher PSNR for a given bits/pixel value because it contains large areas of uniform color.

Next, we compare our technique to another similar technique in the literature. In the graph of figure 3.15, we plot the R-D function for the LENA image for our technique, the simple quadtree technique [30], and the technique used by Shusterman and Feder [40]. From the chart we find that our technique requires more

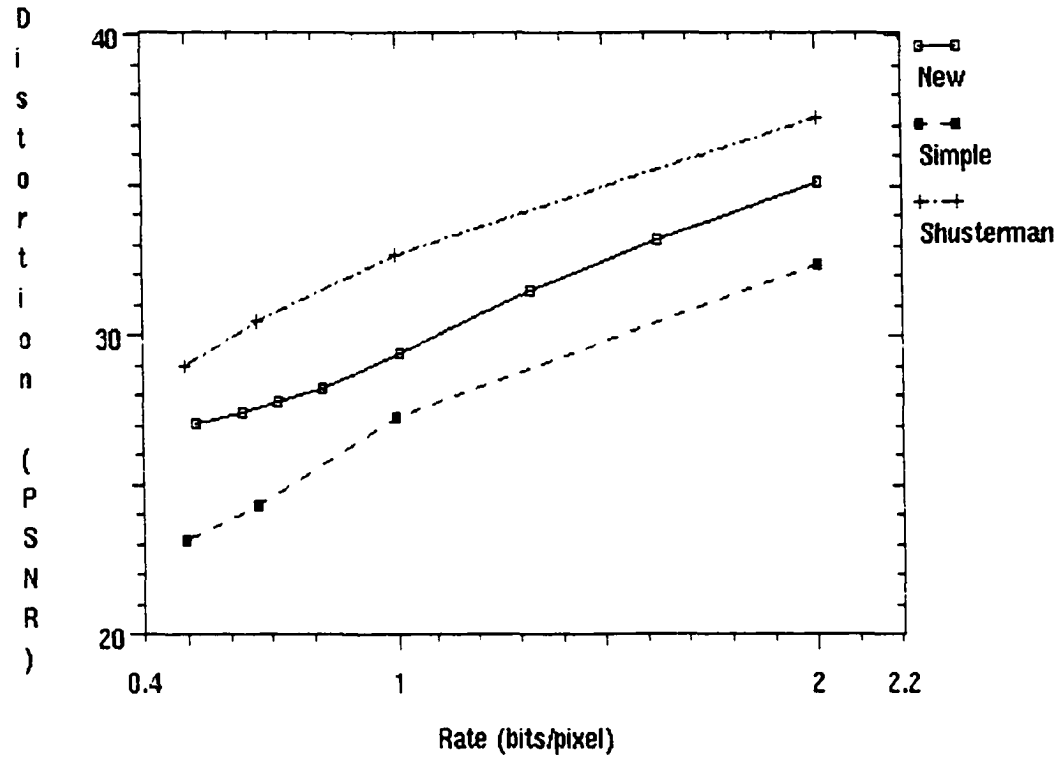


Fig. 3.15. RD function of our Technique and Shusterman's technique, compared to a simple Quadtree structure

bits per pixel than Shusterman's for a certain distortion. Although the Shusterman technique has a better RD ratio, our new technique provides other good features, scalability and error resilience that are shown in the next two sections.

Next, it is interesting to ask: whether the proposed algorithm is competitive with other compression techniques, e.g. DCT-based techniques. To answer this question we compare our algorithm's performance with several popular algorithms described in [3]. The comparison results are shown in figure 3.17, we plot the PSNR values for all encoding techniques of the Lena image at bit rates 0.5, 0.67, 1.0, and 2.0 bits/pixel. The abbreviations of the algorithm names are shown in figure 3.16. The comparison results indicate that the new algorithm, QIET, performs relatively well compared to the others.

Abbreviation	Technique name
QIET	Our new quadtree encoding technique.
SimpleQT	The simple quadtree technique.
SHQT	Shusterman's quadtree-based encoding technique.
Pyramid	Pyramid encoding with scalar quantization.
JPEG	The JPEG technique.
Sub-band	Sub-band encoding with scalar quantization.
SPVQ	Sub-band encoding with pyramid vector quantization.

Fig. 3.16. Abbreviations of the different encoding techniques.

3.2.3 Scalability

The ability of an encoding technique to be decoded at multiple levels of accuracy is very important in applications of computer communications. Most quadtree-based encode only the leaves of the quadtree to obtain a better compression ratio, while losing the favorable scalability and robustness features of the quadtree structure. To test the scalability of our new encoding technique, we performed the following two experiments.

In the first experiment, we encoded the Lena image (512x512) at PSNR 34.136 dB using QIET and Shusterman's technique. QIET required 1.825 bits/pixel, while Shusterman's quadtree required 1.343 bits/pixel. Our representation needs roughly 29 percent extra bits. At the decoder, we scaled down both images by removing nodes from the bottom of the quadtrees, and plot the behavior of both.

Figure 3.18 shows the result of this experiment. Although the Shusterman's encoding required fewer bits to encode the image at a certain quality, it actually required more bits than our encoding when the image was scaled down at the receiver. For example, to scale down the quality to PSNR 27 dB, our encoding required 0.245 bits/pixel while Shusterman's required 0.5 bits/pixel. We see that the new encoding

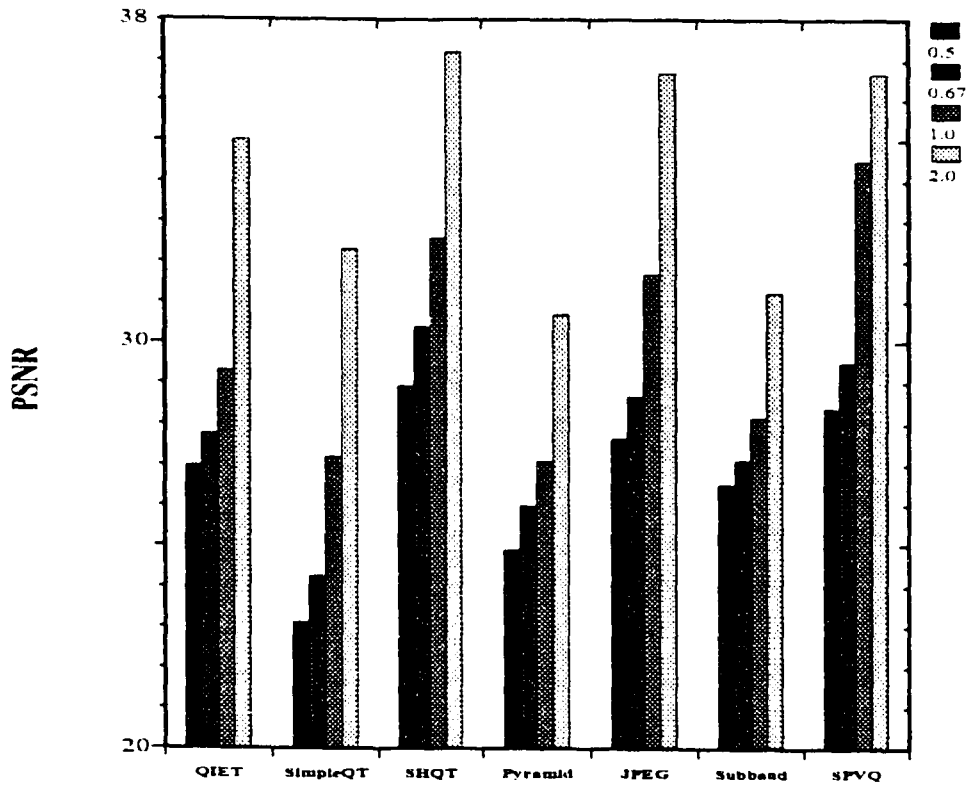


Fig. 3.17. Comparison of several encoding Techniques.

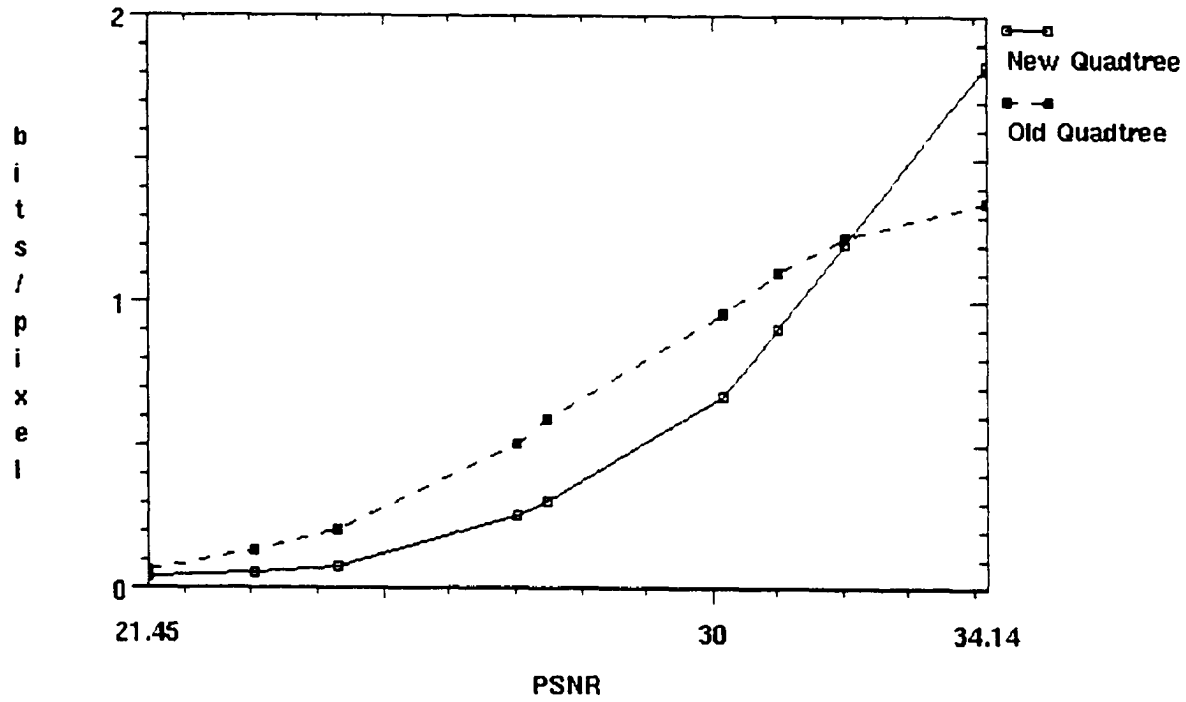


Fig. 3.18. Receiver Scalability, first experiment.

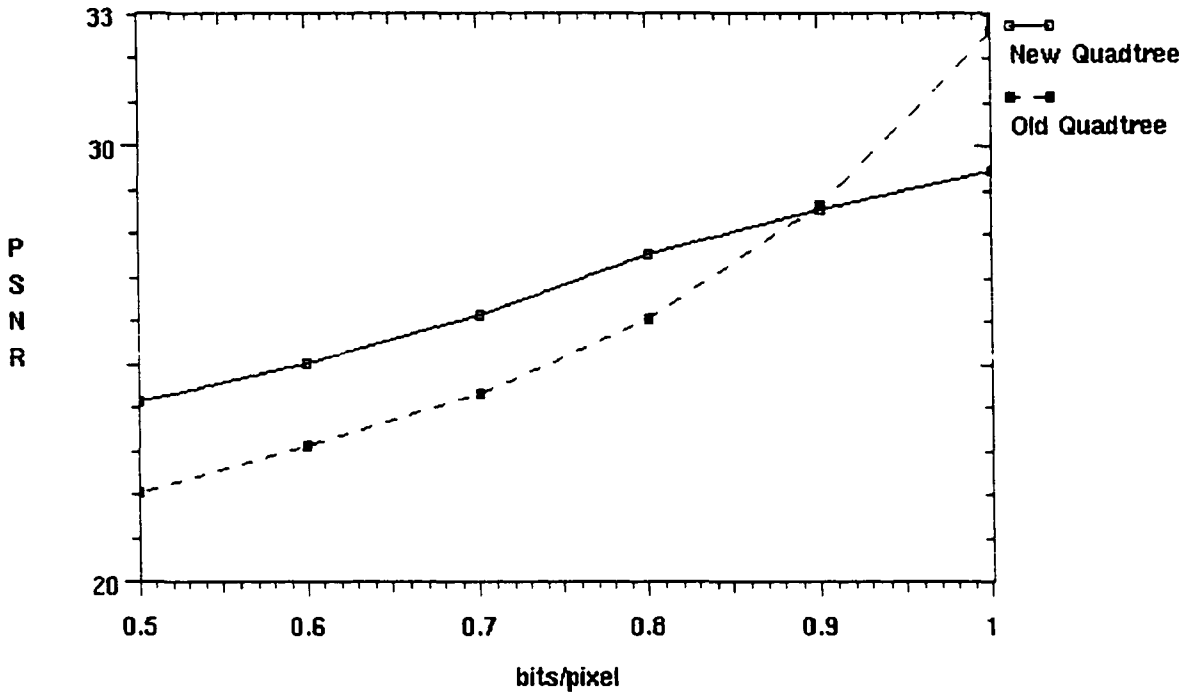


Fig. 3.19. Receiver Scalability, second experiment.

technique scales down more gracefully. We conclude that to scale down an image to a lower PSNR, our encoding always requires less bits/pixel.

In the second experiment, we encoded the Lena image (512x512) at bit rate 1 bit/pixel using both techniques, our encoding gave an image with PSNR 29.3 dB, while Shusterman's quadtree gave an image with PSNR 32.55 dB, an 11 percent improvement. At the decoder, we scaled down both images by removing nodes from the bottom of the quadtrees, and plot the behavior of both.

Figure 3.19 shows the result of this experiment. Although the Shusterman's encoding gave a higher PSNR at the encoder, the PSNR fell sharply when scaled down at the decoder, compared to our algorithm. For example, to scale down the bit rate to 0.7 bit/pixel, our encoding gave a PSNR of 26.1 dB, while Shusterman's encoding gave a PSNR of 24.3 dB. We conclude that to scale down an image to a lower bit rate, our algorithm gives a better quality.

Loss	PSNR of QIET	PSNR of ShQT
0%	32.35	32.35
5%	31.62	27.865
10%	31.1785	27.846
20%	27.27	25.148
40%	23.866	20.461

Fig. 3.20. Decrease in quality due to data loss.

3.2.4 Robustness

A major advantage of our technique QIET is its robustness and tolerance of error. This is due to a number of features. First, QIET is robust because it is based on the quadtree structure, which is by nature error-resilient. Also, we encode and transmit both the leaves and the non-terminal nodes, which is an added redundancy, but enhances robustness. Moreover, in QIET, each node in the tree is an independent unit and is transmitted independent from other nodes, therefore losses do not propagate to other nodes in the same level with different locational codes. Furthermore, the difference field in each node represents the difference between that quadrant and the main quadtree node, not its parent, so any loss does not propagate from father to son node.

Figure 3.20 shows the reaction of our representation and the Shusterman quadtree to random error loss across all levels of the quadtree. We started with the Lena image encoded at PSNR 32.35 dB with both representations, then we randomly dropped a certain percentage of data. We developed a program to read in the data streams and randomly drop the desired percentage of data. The size of the block of data dropped varied randomly from 1 bit to a sequence of 1000 bits. Data was randomly dropped from all the different layers, base and enhancement layers. We reconstructed the image from the lossy data and calculated the new PSNR. As can be seen from the table, our representation is much more error resilient, even at a

large drop percentage. Figure 3.21 shows the resulting image for our technique and Shusterman's at 20 percent and 40 percent data loss.

3.3 Summary

In this chapter, we described a new image encoding technique, QIET, based on the use of quadtrees and vector quantization. The algorithms divide the image into blocks, where the block size is dependent on the image size. Then, we apply a mean-removal algorithm on each block separately. Next, we generate the quadtree for each block and use vector quantization to compress the quadtree data. The resulting encoding is both highly scalable and robust. We compared our algorithm to other algorithms in the literature with respect to image compression, scalability and robustness. It performed well in all three aspects. Thus, the algorithm is attractive for a variety of communication applications over heterogeneous and unreliable networks. In the following three chapters, we describe how we extended the new image encoding technique to represent video sequences.

(a) Original image

(b) QIET, 20% loss

(c) ShQT, 20% loss

(d) QIET, 40% loss

(e) ShQT, 40% loss

Fig. 3.21. Robustness comparison of QIET and Shusterman's technique.

CHAPTER IV

A NEW QUADTREE-BASED VIDEO ENCODING TECHNIQUE (QVET)

In this chapter, we explain how we extended the quadtree-based image representation technique, described in Chapter 3, to enable it to encode video data. We use the well-known IPB technique to achieve a differential encoding that makes use of the temporal redundancy between frames. In this method, the first frame in each GOP is encoded independently, and used as a reference frame. The other frames are encoded differentially, based upon the first frame.

In Section 1, we describe an overview of the system, mentioning only the additions made to the image encoding technique, QIET. We describe the IPB encoding method in detail. In Section 2, we compare QVET with other differential techniques in the literature, and show the experimental results obtained. Finally, Section 3 concludes the chapter.

4.1 Functional description of QVET

The simplest technique to upgrade an image encoding technique to encode video is to treat the video sequence as a set of images and encode each frame independently using the image encoding technique. The advantage of this simple method is in its ease and speed of encoding and decoding, since we do not have to consider relations between frames. Another advantage is robustness, since each frame is encoded inde-

pendently, error does not propagate from one frame to another. However, its major weakness is the low compression obtained because the temporal redundancy between frames, which is particularly high in slow-moving video-conferencing applications, is completely ignored.

We chose to use the differential approach to extend our image encoding technique to video data in order to obtain a reasonable compression result. For robustness, we depend mainly on the nature of quadtrees. Our technique is as follows. First, the sequence of frames is divided into blocks of K consecutive frames, called group of pictures (GOP). The number of frames in the group, K, is an important parameter and directly affects the performance of the system. A large K would typically result in better compression performance due to the large amount of temporal redundancy reduced, but at the cost of less robustness. A small K would result in better robustness, but less efficient compression. A typical value of K is between 6 and 8 frames. The first frame is chosen from the GOP as a reference frame and all other frames are encoded based on the reference frame. The first frame is encoded independently (I-frame). The other frames are divided into a combination of P (prediction) and B (bi-directional) frames. Figure 4.1 shows the dependency between the frames.

This technique is used in most differential video encoding technique. The idea is explained in chapter 2, section 2.3.2. The choice of the sequence of P and B frames affects the performance of the system. Increasing the number of P frames enhances the encoding speed and robustness of the system. Increasing the number of B frames enhances the compression results of the system at a cost of longer encoding and decoding time. A reasonable balance must be made in order to obtain good results, depending on the type of application used.

Figures 4.2 and 4.4 show a block diagram of the encoding and decoding techniques for all frames. We now explain each of these steps in general and not in detail, because they are similar to our new QIET technique, described in detail in Chapter 3.

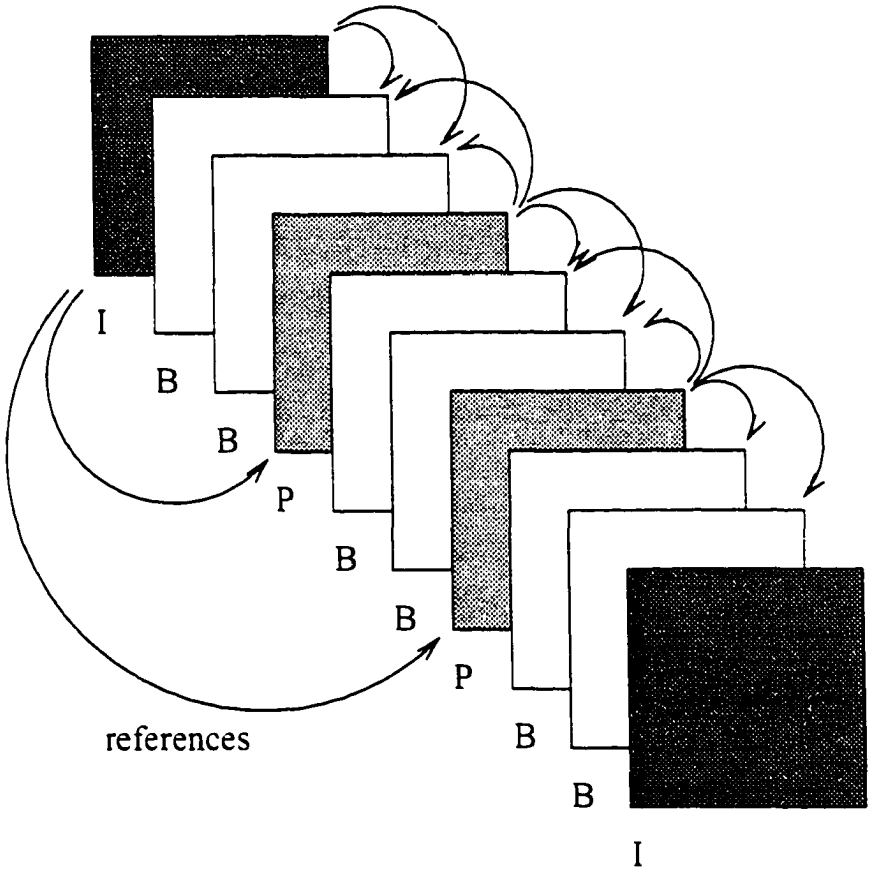


Fig. 4.1. The IPB differential approach to video encoding.

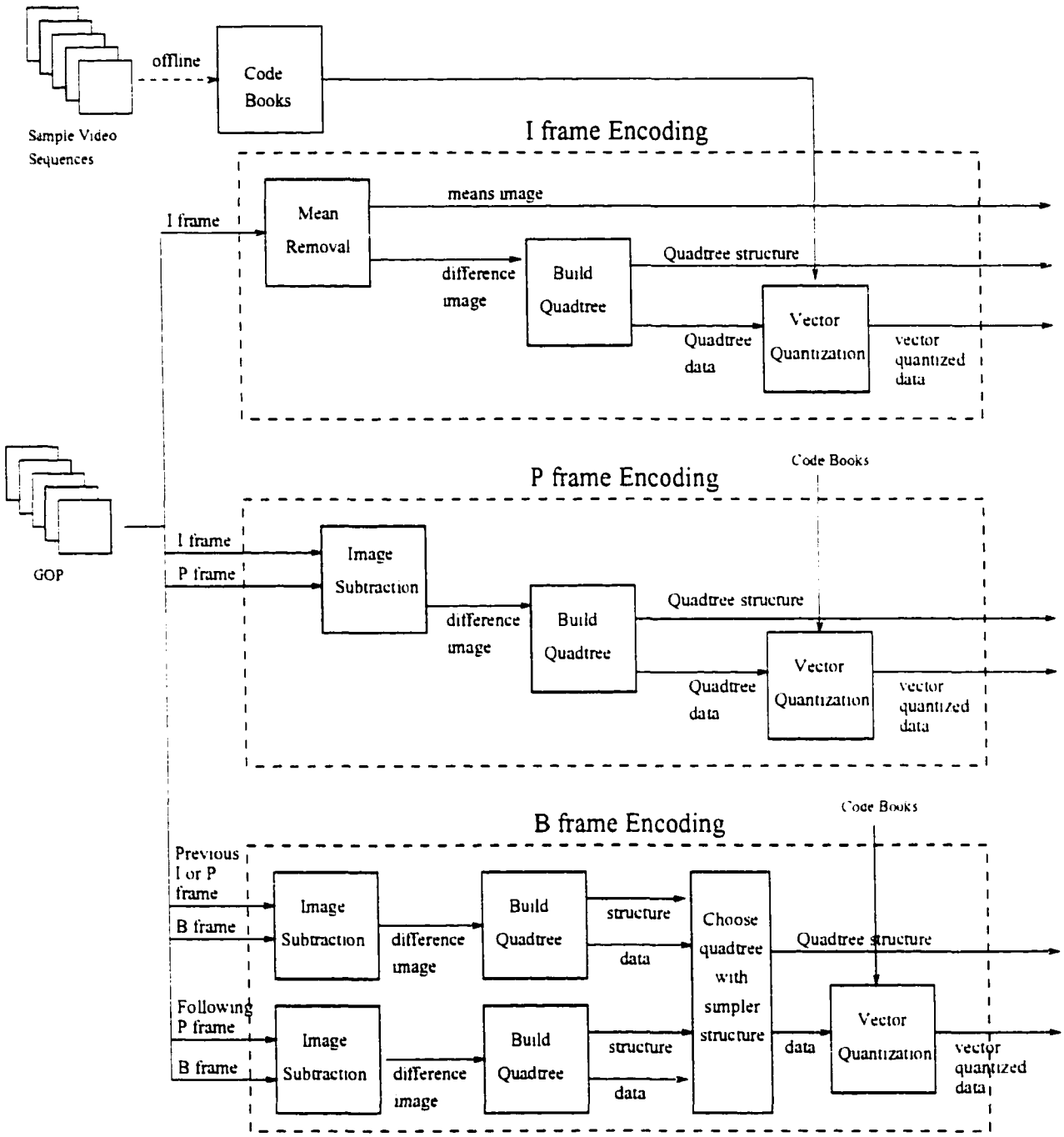


Fig. 4.2. Encoding steps for the different frame types.

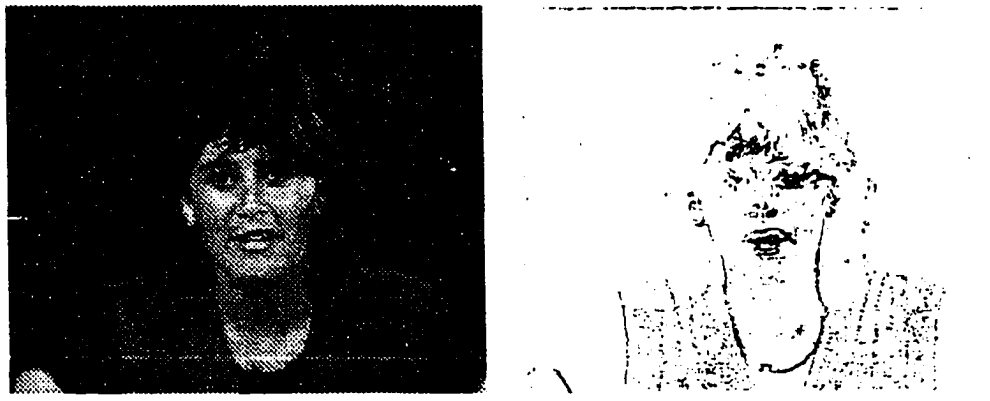


Fig. 4.3. A sample frame and its differential image, relative to the I frame.

For encoding I frames, we start by applying the mean removal technique [31] to the frame to obtain a smoother frame to be encoded, an example is shown in figure 4.3. Then, for each block of the mean-removed frame we obtain the quadtree structure. Finally, the colors of the nodes themselves are encoded using vector quantization. Decoding the I frame is the reverse of encoding: First we decode the vector quantization using the codebooks, then we convert the quadtree structure into array structure. Finally, we add up the means frame to obtain the reconstructed frame. This is identical to the QIET technique for encoding images.

Predictive frames (P frames) are encoded in three steps. First, we generate the "differential" image, which is the difference between the current frame and the previous I frame. The differential frame pixel values would be zero for a pixel that did not differ from the reference frame. For a video conferencing application, the background should result in a mostly blank differential image, and hence good compression. The complexity of this step is identical to that of the mean removal step in QIET.

The next two steps are identical to the QIET algorithm. We convert the differential image into a quadtree. Then we compress it using vector quantization. Note that there is no mean image for P frames. The previous I frame is actually its "base layer", and we cannot decode it without the I frame.

B frames require information of the previous I or P frames and the following P frames (if any) for encoding and decoding. For each block in the frame, we generate the differential image block between it and the previous I or P frame and another one for the following P frame. Then the differential image which generates the simpler quadtree structure, in terms of number of nodes, is chosen to be encoded. Therefore, some blocks in the frame will be dependent on a previous frame, while others are dependent on a forthcoming frame. We need to attach to each block in the frame one bit of information saying whether it should be decoded from a previous or following frame. After that the chosen quadtree is compressed using vector quantization.

The B frames have the highest compression ratio, but they also have the highest encoding and decoding complexity. They also require out-of-order processing of frames at the decoder, because a B frame cannot be decoded unless the following P frame has been decoded. And thus, they need a large buffering capacity.

The nature of the sequence of I, P, and B frames will be determined by the application requirements. For more robustness, the best choice would be to have a large number of I frames (hence shorter GOPs) to decrease inter-frame dependency and error propagation. For fast encoding and decoding, the best choice would be to abandon using B-frames altogether because they require much more encoding time than I frames or P frames. For best compression results, we should use longer GOPs and more B frames.

The decoding process of I frames is identical to that of QIET, as shown in figure 4.4. For decoding P frames, we use the previously decoded I frame as a base layer, instead of the mean image. Otherwise, the technique is similar to the I frame.

Decoding of the B frame is more complex and time consuming. First, it requires that the following P frame (if any) be decoded first, so it has to be buffered for a while. Then for each individual block in the frame, we perform reverse vector quantization, and conversion from quadtree into image array. We add up the differential block to the decoded block of the previous or following I or P frame.

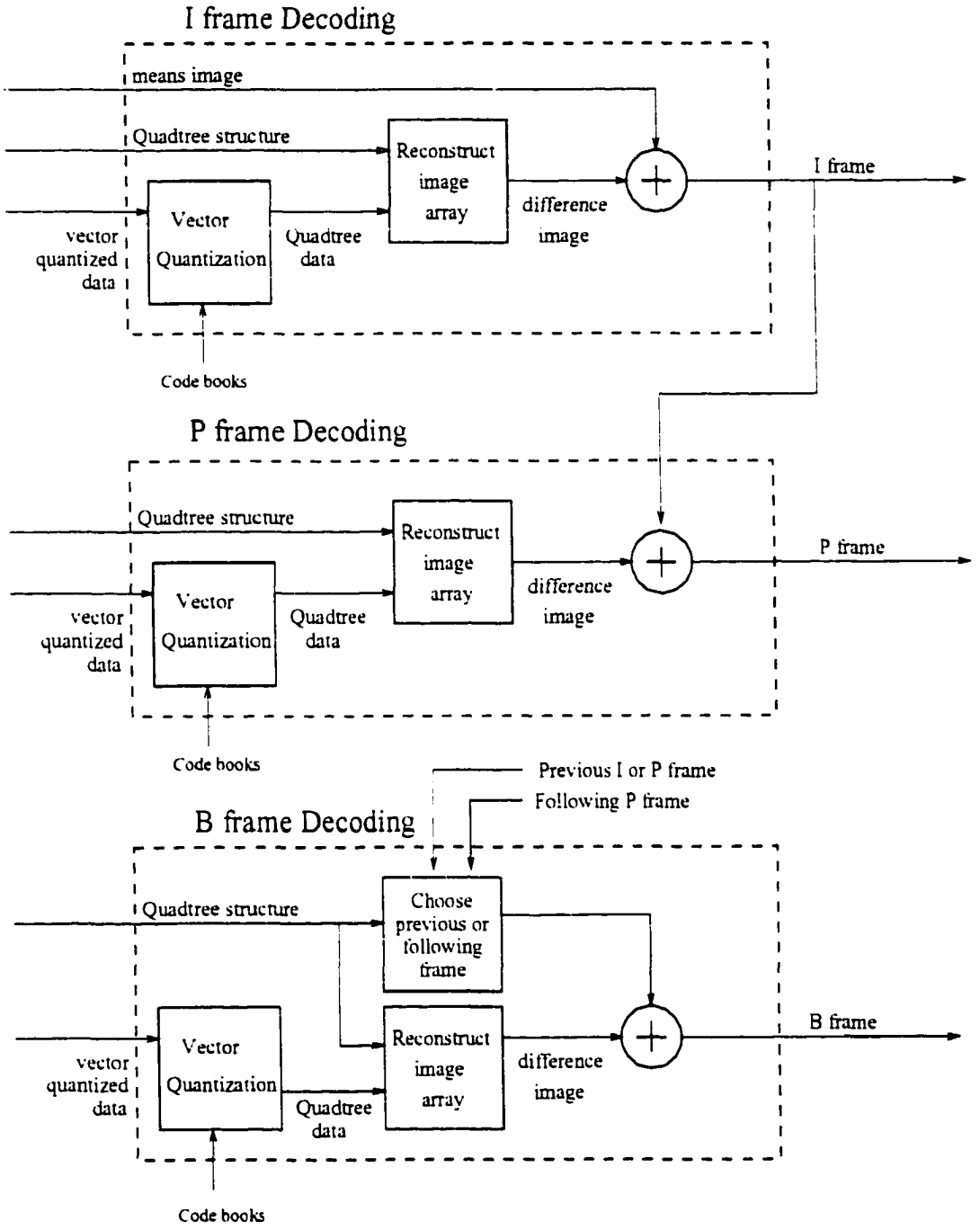


Fig. 4.4. Decoding steps for the different frame types.

depending upon which one was used during the encoding process.

4.2 Analysis and Results

In this section, we show the results of our new encoding technique and compare it to the MPEG algorithm [31]. We compare them according to their ability to compress an image, scalability, and robustness. We use the Peak-Signal-to-Noise ratio (PSNR) as our measure of quality, as described in section 3.2.1.

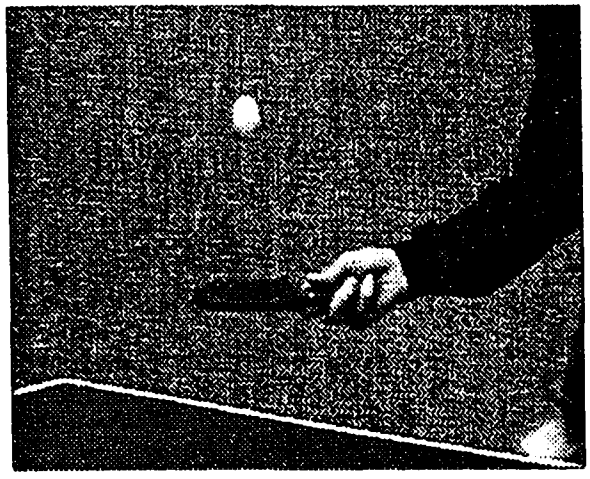
4.2.1 Compression Results

The proposed video representation technique was tested on a set of three standard gray-scale video sequences used for testing encoding techniques. The size of the source pictures is 176x144 and the first frame of each one is shown in figure 4.5: Miss America (MISSA), Table tennis, and Football.

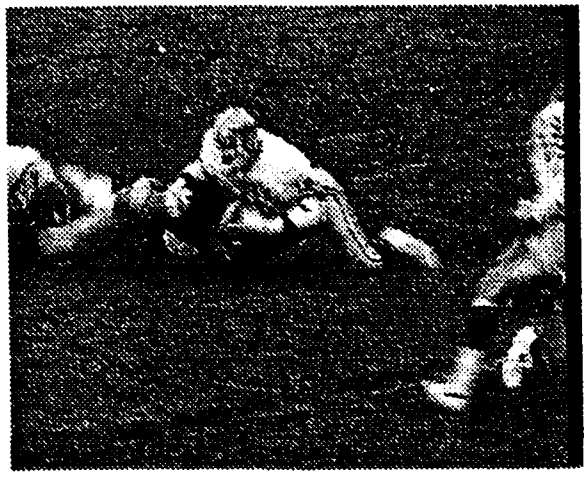
Each sequence has different characteristics. MISSA is a video sequence of a talking person with few movement, which is typical in video conferencing applications. Table Tennis is a video sequence with a fast-moving object (the ball) and a large static background. Football is the most challenging sequence because of its very high speed and low temporal redundancy. In this section, we show the results of our experiments with the MISSA sequence only, as our system is targeting the field of video-conferencing, where it is a standard benchmark for testing and comparing systems.

In the graph of Figure 4.6, We plot the PSNR of the first 10 frames of Miss America for three different GOPs: All I frames, IPPPPP repeatedly, and IBPBPB repeatedly. We keep the average bit rate at 16kbps, and 10 frames/sec. We chose to keep the bit rate constant and measure the resulting quality of the encoded video sequence. Another option was to keep the quality fixed and measure the needed bit rate to achieve that quality. However, this option is more difficult to implement and experiment with.

(a) Miss America (MISSA)



(b) Table Tennis



(c) Football

Fig. 4.5. First frame of the three test sequences used.

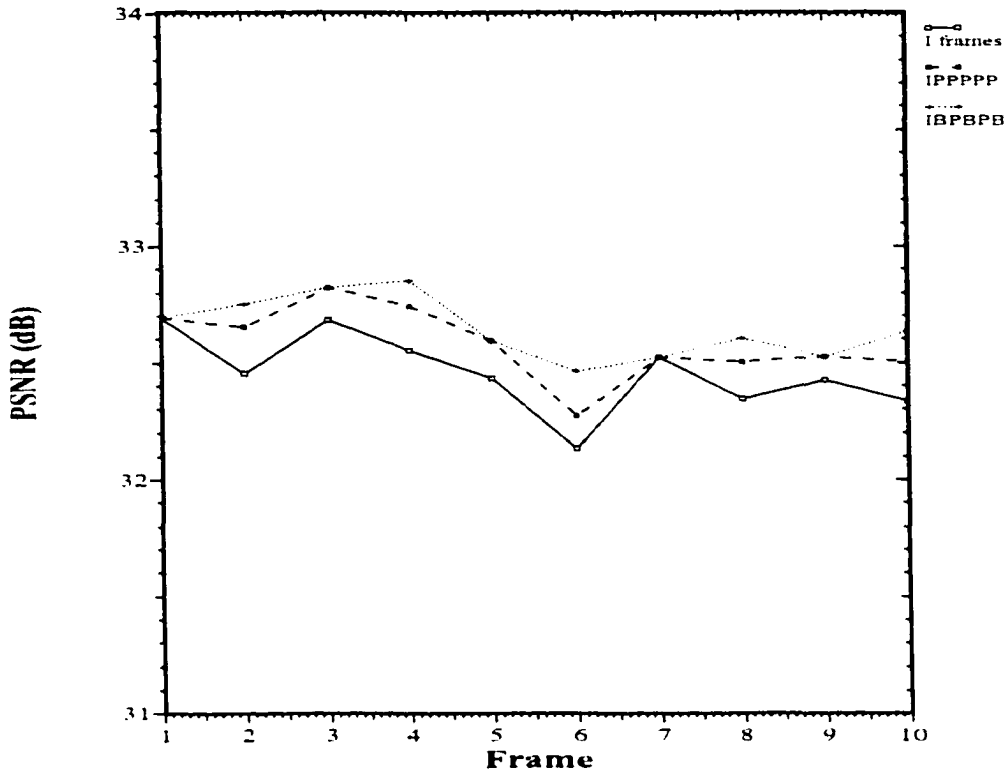


Fig. 4.6. PSNR for different GOPs of the Miss America sequence.

From the figure, we see the compression advantage of using P and B frames is clear. By keeping the total encoded image size constant, we obtain a better quality image, than I frames. The P frames obtains a large improvement in quality. This is mainly due to the fact that it generates smaller quadtree since the differential image is mainly blank. The B frames obtain only a modest improvement over the P frames. This is due to the fact that B frames demand more processing time, which shadows its compression enhancements.

Next, we compare our technique to other similar techniques in the literature. In the graph of figure 4.7, we plot the PSNR function for the Miss America sequence

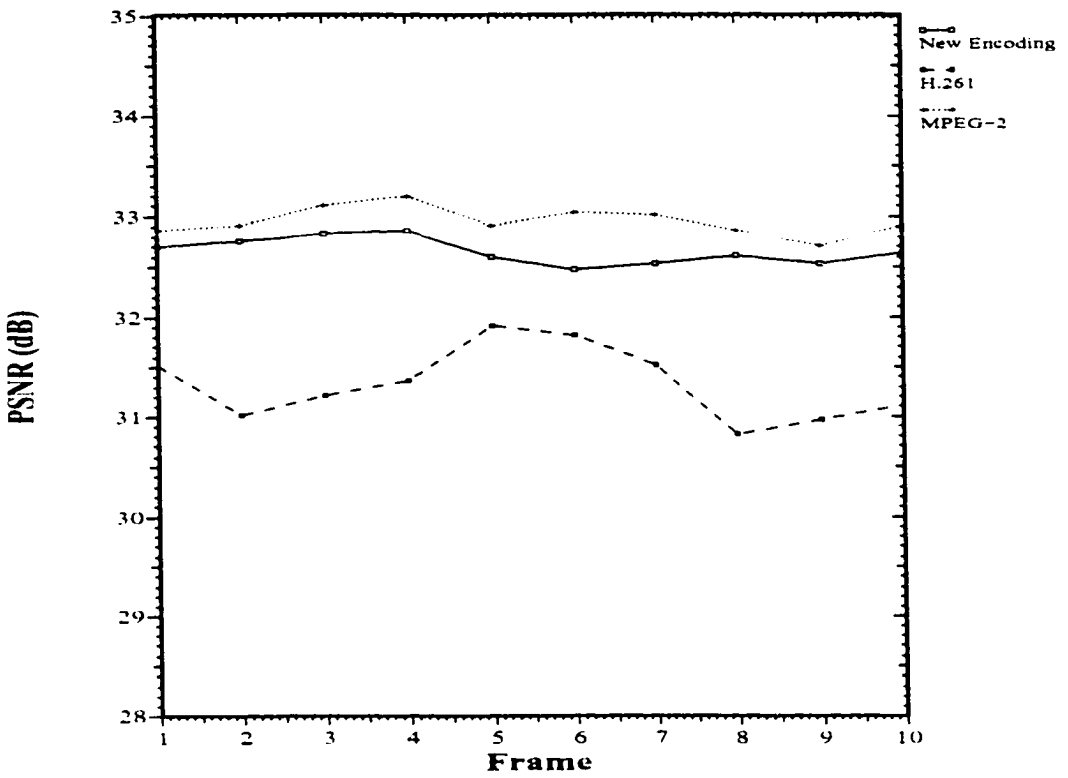


Fig. 4.7. PSNR for different encoding of the Miss America sequence.

for QVET, the H.261 protocol, and MPEG2. The average bit rate is 16kbps at 10 frames/sec for all techniques. We used the GOP (IBPB) for our sequence and MPEG2. From the chart, we find that our technique gives a slightly lower quality than MPEG-2, for a certain bit rate (but much better than H.261). However, our new technique provides other good features, scalability and error resilience, which overcome this drawback. We will explain this in the next two sections. We obtained similar results with the other two test video sequences, Football, and Table Tennis, as shown in figures 4.8 and 4.9, respectively.

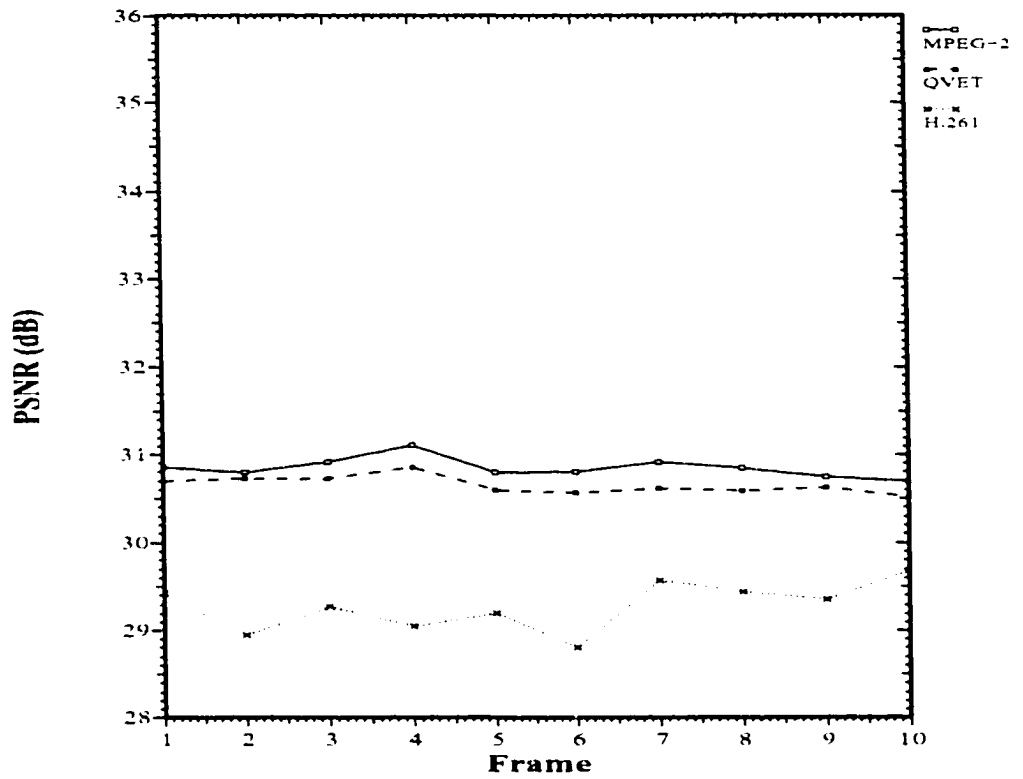


Fig. 4.8. PSNR for different encoding of the Football sequence.

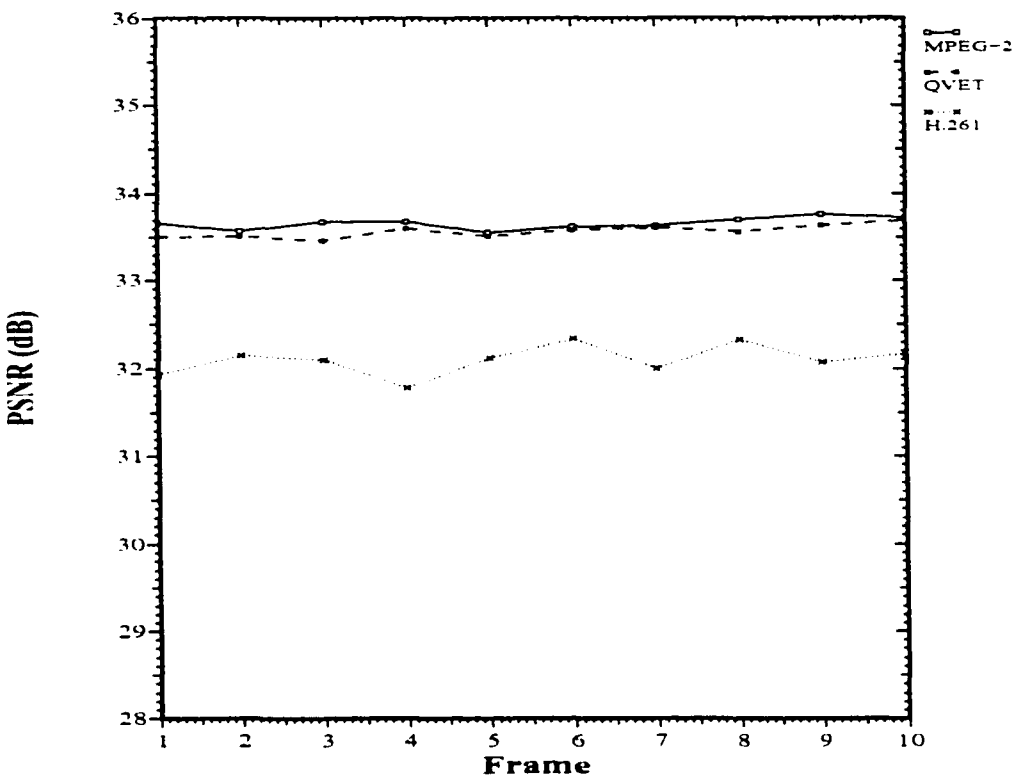


Fig. 4.9. PSNR for different encoding of the Table Tennis sequence.

4.2.2 Scalability

The ability of an encoding technique to be decoded at multiple levels of accuracy is very important in applications of computer communications. Our algorithm is scalable because all nodes of the quadtree are encoded, leaf nodes are encoded implicitly inside their parent non-terminal nodes, as explained in detail in chapter 3. To test the scalability of our new encoding technique, we performed the following experiment.

We encoded the first frame of the Miss America video (176x144) at bit rate 16 Kbps using QVET and MPEG-2, which is scalable, too. QVET gave an encoded frame image with PSNR 32.69 dB, while MPEG-2 gave an image with PSNR 32.85 dB, a slightly better improvement. At the decoder, we scaled down both frames to different sizes and plotted the behavior of both. QVET is scaled down by either dropping entire layers of the quadtree, which gives a dramatic scale down. Or, for finer down-scaling, we can intelligently trim down only parts of the quadtree. The experiment was repeated with several other frames with almost identical results.

Figure 4.10 shows the result of this experiment. Although the MPEG-2 encoding gave a higher PSNR at the encoder, the PSNR fell sharply when scaled down at the decoder, compared to QVET. For example, to scale down the total frame size to 1Kbits (from 1.6 Kbits)by decreasing the quality of the decoded frame, QVET gave a PSNR of 29.06 dB, while MPEG-2 encoding gave a PSNR of 27.51 dB. We conclude that to scale down a frame to a lower bit rate, our algorithm gives a better quality due to the naturally scalable structure of the quadtree.

4.2.3 Robustness

Robustness in QVET is mainly based on the quadtree structure. Robustness is also affected by the choice of sequence of P and B frames in the GOP. The less B frames in the GOP, the better the robustness. Also, shorter GOPs result in better robustness. This is an inherent feature in all differential video-encoding systems.

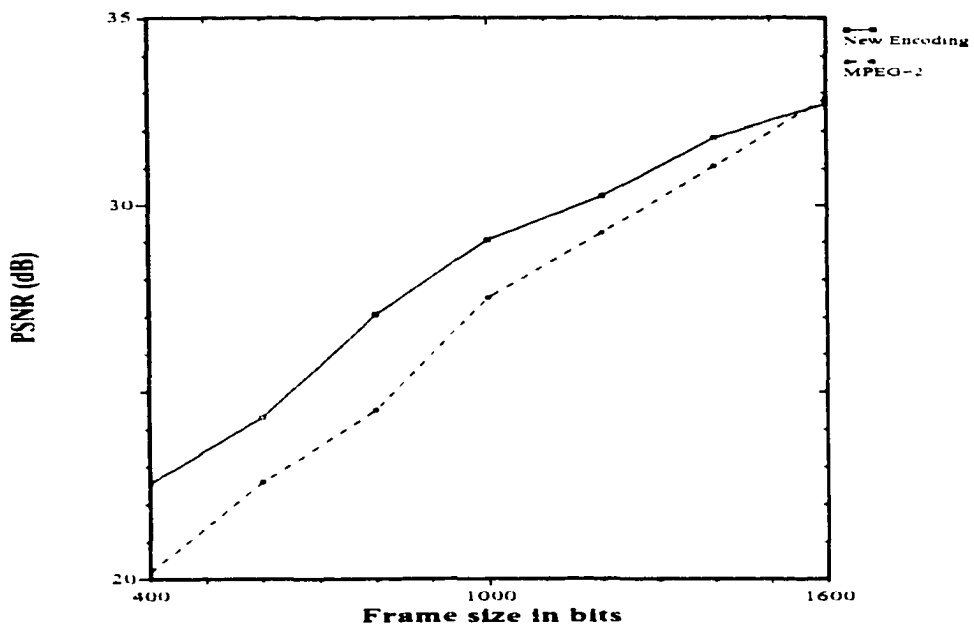


Fig. 4.10. Scalability results for the first frame of Miss America Sequence.

Loss	PSNR, QVET	PSNR, MPEG-2
0%	32.69 dB	32.85 dB
5%	32.03 dB	30.67 dB
10%	31.17 dB	28.14 dB
20%	27.27 dB	23.07 dB
30%	24.62 dB	20.83 dB
40%	23.86 dB	18.47 dB

Fig. 4.11. Decrease in frame quality due to data loss.

Figure 4.11 shows the reaction of our representation and the MPEG-2 technique to random error loss across all levels of the quadtree. We encoded the first frame of the Miss America video (176x144) at bit rate 16 KBPS using QVET and MPEG-2. QVET gave an image with PSNR 32.69 dB, while MPEG-2 gave an image with PSNR 32.85 dB. Then we randomly dropped a certain percentage of data, and reconstructed the frame from the lossy data and calculated the new PSNR, as we did for QIET in chapter 3. As can be seen from the table, our representation is much more error resilient.

4.3 Summary

In this chapter, we described how the quadtree-based image representation technique, QIET, was extended to encode video. we developed a new encoding technique, QVET, based upon it. To make it a differential technique and make use of the temporal redundancy between frames, we applied the well-known differential IPB technique to QIET.

We obtained good scalability and robustness properties, mainly due to the nature of the quadtree. And we obtain reasonable, if not dramatic, results in the compression aspect. Our technique also proved to be more scalable and robust than

the MPEG2 technique.

However, this method of video encoding could be improved upon to obtain better compression results while keeping the same theme of the encoding technique. Instead of adding the differential method to the image encoding, we can extend the image encoding itself to represent video. We use the octree, which is the three-dimensional equivalent of the quadtree, to represent the GOP. This is a more natural and simple approach to video encoding.

CHAPTER V

AN OCTREE-BASED VIDEO ENCODING TECHNIQUE (OVET)

We now explore another method of extending our image encoding technique to encode video streams. The basic idea here is to use exactly the same technique and replace the quadtree structure with an octree structure. The octree is the three-dimensional equivalent of the quadtree, and is typically used to represent three-dimensional objects. In this chapter, we show how it can also be used to successfully encode video as well. We simply treat each GOP as a three-dimensional object, and not a sequence, and encode them collectively as one item. We also combine both quadtree and octree techniques to generate a new encoding having better characteristics.

In section 1, we describe the octree representation method. In section 2, we describe the new octree-based video representation technique. Section 3 shows the experimental results obtained. We obtained a great improvement in compression, as well as scalability and robustness. In section 4, we describe another video encoding technique, the hybrid video encoding, which has a differential approach and uses both the quadtree and octree representations. We also show the experimental results obtained by it, in compression, and robustness. Finally, section 5 concludes the chapter.

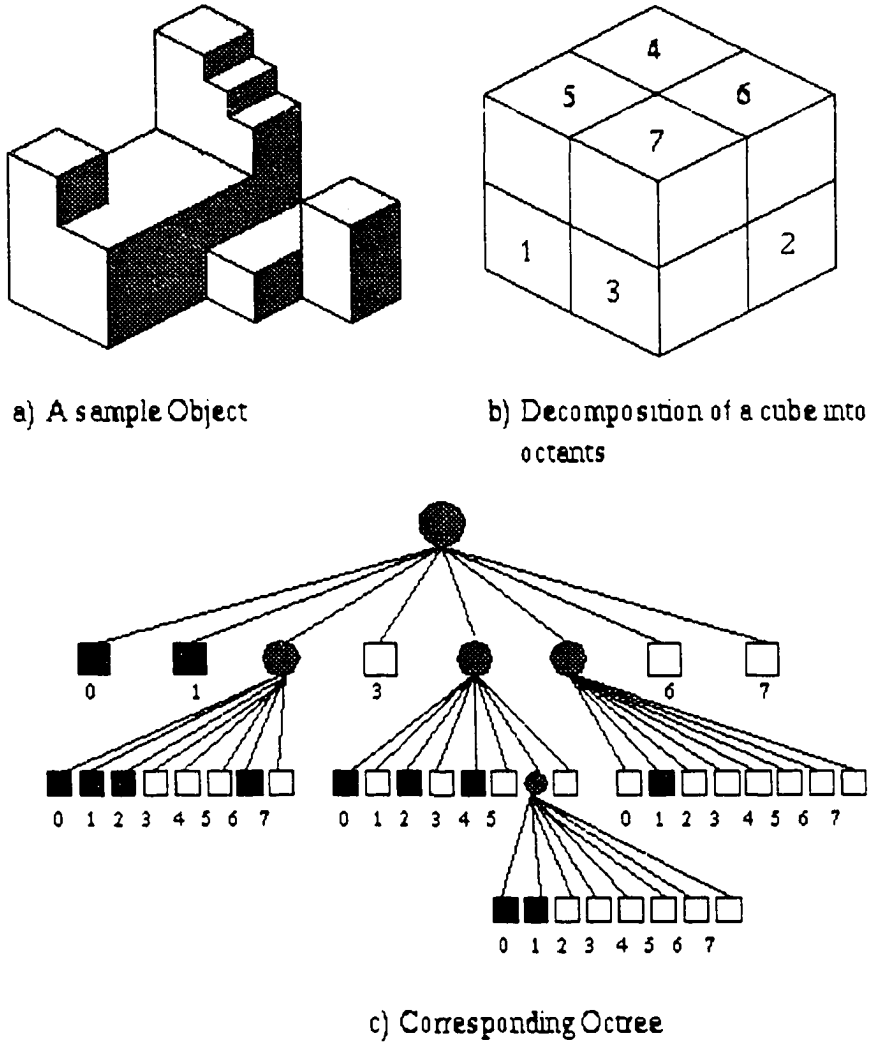
5.1 The Octree Representation

The octree is an approach to object representation based on successive subdivision of an object into sub-objects. It is typically used in robotics and object manipulation fields [38]. In this dissertation, we show that it can also be used successfully in video encoding. The basic idea here is to treat the video sequence as a 3D object with the time axis representing the depth of the object, as shown in figure 5.2. The video sequence is divided to blocks where each block consists of a certain number of frames to be encoded together as an object. We propose to use the octree [38, 39] technique to encode this object. The number of frames in each block defines the "depth" of the octree, while the frame size defines the height and width.

An octree is the three-dimensional equivalent of a quadtree. Octrees representation is based on cubic decomposition of space. We repeatedly subdivide the object into 8 octants (sub-blocks) and examine each octant in turn for homogeneity in color. This is repeated until we obtain blocks (possibly single voxels, a cubic minimum unit with equal width, height, and depth) which consist entirely of only one color. This process is represented by a tree of out-degree eight (and hence the name) in which the root node represents the entire object, the eight sub-octants represent the eight corners of the objects, and the leaves (terminal nodes) correspond to those blocks of the object for which no further subdivision is necessary.

Figure 5.1 shows a sample object and its octree. We chose a simple black and white object for simplicity, but the same technique applies for color objects. The resulting tree is of depth three. Circular nodes (non-terminals) in the tree mean that this part of the object is of different colors and will be decomposed further. Usually, the average color of that area is stored in this node. A black node represents a black part of the object. A white node represents a white part of the object.

The most obvious storage structure for an octree is a fully pointered tree [30] where each node record has 8 pointers to its eight sons, and a 'color' field representing the node's color or the average color of its 8 sons if it is a non-terminal node.



a) A sample Object

b) Decomposition of a cube into octants

c) Corresponding Octree

Fig. 5.1. Octree structure.

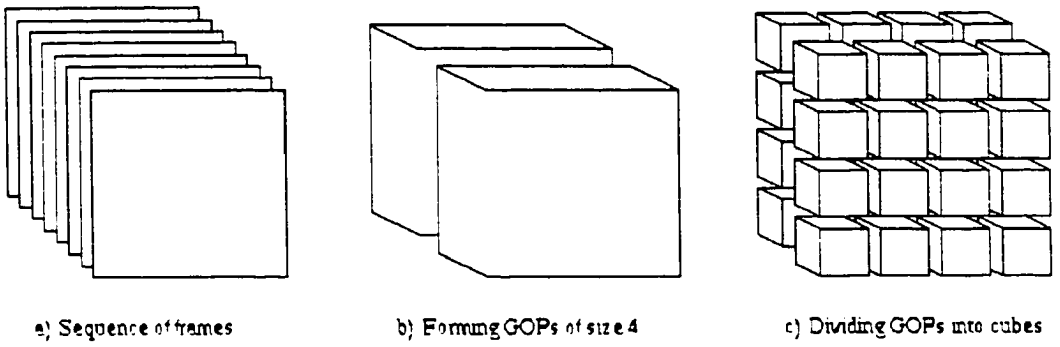


Fig. 5.2. Conversion from frames into cubes.

Although this method is simple and easy to implement, it needs a lot of storage space. Other octree representations has been proposed : Gray-Code [39], Tree-Code [25], DF-octree [16], and Goblin octree [47]. In developing this technique, we use the modification of the Gray-code, similar to that of Chapter 3.

5.2 An Octree-based Video Encoding Technique, OVET

The new encoding technique is based upon our earlier work on image compression, described in chapter 3. The technique can be summarized as follows:

First, the sequence of frames is divided into blocks of K consecutive frames, called group of pictures (GOP). The value K should be a factor of two to simplify the building of the octrees. 8 or 16 should be a good choice. The GOP is then looked upon as a three-dimensional cuboid and encoded as an object, with 3 dimensions, the width and height of the frames, and the time factor. Each pixel becomes a "voxel". The cuboid is then divided into an array of cubes, each to be encoded separately. At the decoder side, the reverse of this division is done. The frames are once again re-constructed from the set of cubes. Figure 5.2 shows this abstraction.

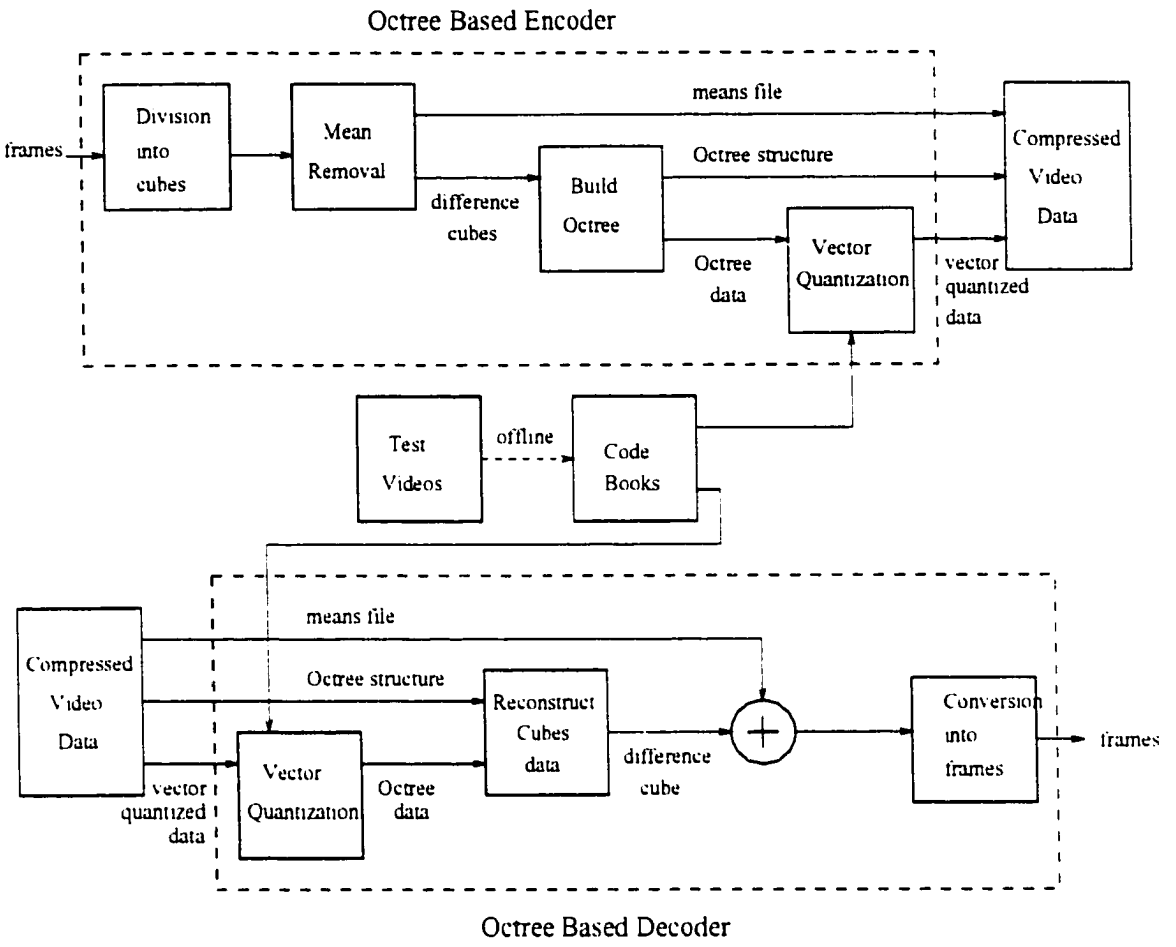


Fig. 5.3. Block diagram of the Octree Encoding Technique.

The next steps are as follows: we apply a mean removal technique to each cube in turn, generating "difference cubes" and then these cubes are converted to a forest of octrees. Next, the octree data is compressed using vector quantization and a set of previously built codebooks. The vector quantization codebooks are built off-line using a set of training images. The decoding process is symmetric to the encoding process but it is less time consuming because converting the octrees back to pixels is less time-consuming than building the octree, and we do not perform a search in the codebooks. Figure 5.3 shows a block diagram of the encoding and decoding process. In the remainder of this section we explain each of these steps in detail, and show the algorithm used.

5.2.1 Mean Removal

The first step in encoding a GOP is to remove the means of each (nxnxn) block. We first calculate the means of each (nxnxn) block, and then subtract the value of the mean from each pixel in that block. The mean and the mean-removed (difference) cubes are then encoded separately. The main issue here is how to choose the block size (nxnxn). If the block size is relatively small, then the compression will not be efficient, as the mean block size will be too large, and the block identifier will need more bits and the resulting octree will be too shallow to provide useful scalability. If the block size is relatively large, then a larger octree will be needed, and consequently more errors will be unrecoverable.

We experimented with three different block sizes: 4, 8, and 16. The advantages of the bigger size blocks is that we got a better compression ratio for a slow-moving video sequence, and a more scalable code since we will have a larger octree. But the compression will be less robust because an error can affect a larger number of frames. On the other hand, the 4x4x4 block size gave the worst compression ratio, but was the most robust compression. Since we are targeting video conferencing applications, we chose to have a large block size 8x8x8, to obtain good compression and still maintain reasonable robustness.

5.2.2 The Octree Construction Algorithm

In this section we explain how the octree structure is build from the difference block. The algorithm is similar to the quadtree building algorithm, described in section 3.1.2. The only difference is that we need to load the whole block of frames first, before we can start building the octree. The algorithm is shown in figures 5.4 and 5.5.

The execution time of this algorithm is proportional to eight-seventh the number of pixels in the entire block since it is the number of times procedure CONSTRUCT is invoked (and equal to the number of nodes in a complete octree for

```
procedure BUILDOCT(BLOCK. LEVEL)
/* construct a octree encoding for the 2^LEVEL x 2^LEVEL x 2^LEVEL block in pixel
array BLOCK. The origin is assumed to be the back SE-most pixel of the block */
begin
  Load the frames required to form the block.
  Remove the means from the block and encode separately.
  preload global integer array XF[ ] with 1.0.1.0.1.0.1.0 :
  preload global integer array YF[ ] with 1.1.0.0.1.1.0.0 :
  preload global integer array ZF[ ] with 1.1.1.1.0.0.0.0 :
  preload global character array CD[0.1.2.3.4.5.6.7] with '0','1','2','3','4','5','6','7' :
  set CODE = NULL string :
  result = CONSTRUCTOCT(LEVEL,2^LEVEL, 2^LEVEL, 2^LEVEL, CODE);
  return;
end
```

Fig. 5.4. Octree Encoding Algorithm.

```
integer procedure CONSTRUCTOCT (LEVEL, X, Y, Z, CODE):
/* Generate the code corresponding to the 2^LEVEL x 2^LEVEL x 2^LEVEL
portion of the pixel array BLOCK whose back SE-most pixel is BLOCK[X,Y,Z] */
begin
  integer array RES[8] : integer AVERAGE, DEV :
  if (LEVEL == 0)
    then return (BLOCK[X,Y,Z]) /* At a pixel */
    else
      begin
        LEVEL = LEVEL - 1; D =2^LEVEL ;
        for i=0 to 7 do
          RES[i] = CONSTRUCTOCT(LEVEL,X-XF[i]*D,
                                Y-YF[i]*D,Z-ZF[i]*D,CODE+CD[i]);
        AVERAGE = average(RES[0].....RES[7]);
        DEV = Maximum absolute deviation from the average ;
        if (all sons are leaf nodes) and (DEV < Threshold)
          then
            return (AVERAGE)
          else
            begin
              Search the codebook to find the nearest match to the
              colors of the eight color fields.
              Create a data structure for the current node, which
              includes the Codebook index and the Block ID.
              Store the types of the eight sons in the octree
              structure list (0 for leaf, 1 for non-terminal node).
              return (AVERAGE)
            end
      end
end
```

Fig. 5.5. Octree construction algorithm.

a 2^{LEVEL}x2^{LEVEL}x2^{LEVEL} block). The algorithm is recursive and the maximum depth of recursion is equal to the log of the block size (3 for an 8x8x8 block). Due to temporal redundancy between frames, the complexity of building one octree for eighth frames, was experimentally found out to be about a third of the complexity of building eight independent quadtrees for the different frames.

5.2.3 Block Representation

After mean removal, we convert the difference blocks into octrees. The "Gray Code" octree representation method was first described in detail in [39]. We used it in QIET for encoding images, and we will use it here to encode octrees in a similar fashion. A brief description of this representation follows.

The Gray-Code method stores the octree as a group of several lists, each list representing one level of the octree. In each list we store only non-terminal (Gray) nodes. Each node record contains the node's locational code, and 8 fields containing the average color of each of its 8 sons. The locational code specifies both the size and the exact location of the octant in the block.

The OVET representation is based on the Gray-Code with some modifications to obtain better compression ratios. Since we use blocks of size (nxnxn) voxels, the resulting forest of octrees will each be $log_2(n)$ levels deep. Each octree represents a volume of size (nxnxn) of the GOP. We start by sending the octree structure of each octree in turn. Its locational code and a series of bits that show its octree structure represent each tree. We indicate termination by a leaf with a '0' and branching into child nodes with a '1'. For example the octree of figure 5.1 is represented by 1 00101100 00000000 00000010 00000000 00000000. Since the lowest level consists of all leaves we do not need to represent it. Figure 5.6 shows the quadtree structures for a sample 8x8x8 block.

Clearly, the octree structure is of very high importance, since without it we cannot put an incoming node in its correct location. Therefore, we choose to send

Block	Level	Structure
10	3	1
	2	00101100
	1	00000000 00000010 00000000
	0	00000000

Fig. 5.6. Octree structure for a sample block.

the octrees structure first separately, before sending the rest of the information.

5.2.4 Vector Quantization

We apply vector quantization to each level of the octrees separately, since the contribution of each node to the frame area is proportional to the size of the node, which is determined through its level. Therefore, more accuracy is needed for nodes at higher levels compared to nodes at lower levels.

The locational code of the octree and the colors of its eight sons represent each octree node. The locational code specifies the location of this node. The colors of the eight sons represent the average colors of the eight octants of this node. Since most pixel colors are actually close to the mean color, we can quantize the colors to use fewer bits with a low level of distortion. We use vector quantization to quantize all eight color fields together, making use of the spatial locality between them and using fewer bits to encode them.

Vector quantization is based upon the use of a codebook. Since all levels of the octree are not of the same importance, we use a different codebook, with a different size, for each layer. The codebook size reflects the level of detail of a layer, the higher a layer, the larger its codebook becomes (relative to the number of nodes in that layer). We chose a codebook of size 64 items for the lowest layer (6-bit index) and double the size for each higher layer.

5.2.5 Decoding Algorithm

The decoding algorithm is the reverse of the encoding algorithm. However, it is much more time-efficient. The decoding steps are:

- Start by generating our basic block of frames by enlarging the mean block to the actual frame size. For example, if our block size is (8x8x8), then each pixel in the mean image would be enlarged to cover a cubic (8x8x8) area over the 8 frames to be decoded together.

- Perform reverse vector quantization to obtain back the octree node color data. The receiver has the same codebooks as the sender, and uses them to find the data vectors from the codebook indices. This is a direct lookup in the codebooks, not a search like in the encoding process.

- Given the octree node color data, and the octree structure, re-build the octree structure. Only as many levels as the receiver can process should be decoded. Since the levels are processed in a decreasing order, we can build the octrees of each block as we process the nodes. Each level gives information about the following level's data structure.

- As the quadtree is being decoded, we plot the data obtained directly on the basic block to enhance it, in a layer by layer approach. Thus, we end up with 8 decoded frames, and they are displayed in turn. This algorithm is explained in detail in the following subsection.

Octree Destruction Algorithm

The decoding algorithm is similar to that of the quadtree decoding algorithm, described in section 3.1.5. It is shown in figure 5.7. The main addition is that we plot the decoded data over the set of frames included in the GOP.

The execution time of this algorithm is proportional to one-seventh the number of pixels in the block, since it is the number of node data received (and equal to

```
Read means block:
Read Octree structure:
Load required codebooks:
set OTREE pointer = null:
Display Means block in all the frames, enlarged to the real frame size.
For each required level do
  begin
    Read node data, INDEX
    Get the 8 octants color using the INDEX and the codebooks.
    Get the Actual location of the eight octants from the OTREE pointer:
    Add the mean value of this block to all eight colors to get
    the actual octant color.
    Plot the octants in their positions in the different frames.
    Move the OTREE pointer to the next expected node to arrive.
  end
```

Fig. 5.7. Octree Decoding algorithm.

the number of **GRAY** nodes in a complete octree for a 2^{LEVEL}x2^{LEVEL} x2^{LEVEL} block).

5.3 Analysis and Results

In this section, we show the results of our new encoding technique and compare it to the MPEG algorithm [31]. We compare them according to their ability to compress a video, scalability, and robustness.

5.3.1 Frame Size Complexity

The presented video representation technique was tested on our set of three standard gray-scale video sequences used for testing encoding techniques, Miss America (MISSA), table tennis, and football. The size of the source frames is 176x144. This frame size results in an array of size (22x18) blocks, each block is a cube of dimensions (8x8x8).

In the graph of figure 5.8, we plot the PSNR function for the Miss America sequence for OVET, the QVET described in chapter 4, the H.261 protocol, and MPEG2. The average bit rate is 16kbps at 10 frames/sec, a typical rate used in video conferencing. From the chart, we find that our technique gives a much better quality (for a fixed bit rate) than the other techniques. This result is due to the fact that we are encoding the frames as three-dimensional blocks and therefore can have better compression due to the temporal and spatial locality. Figures 5.9 and 5.10 show the results of the same experiment with the football and table tennis test sequences, respectively.

The performance of each technique differs according to the sequence used. We notice from the graphs that all techniques perform best for the slow-moving sequence MISSA. This is due to the fact that slow-moving sequences contain a lot of temporal redundancy.

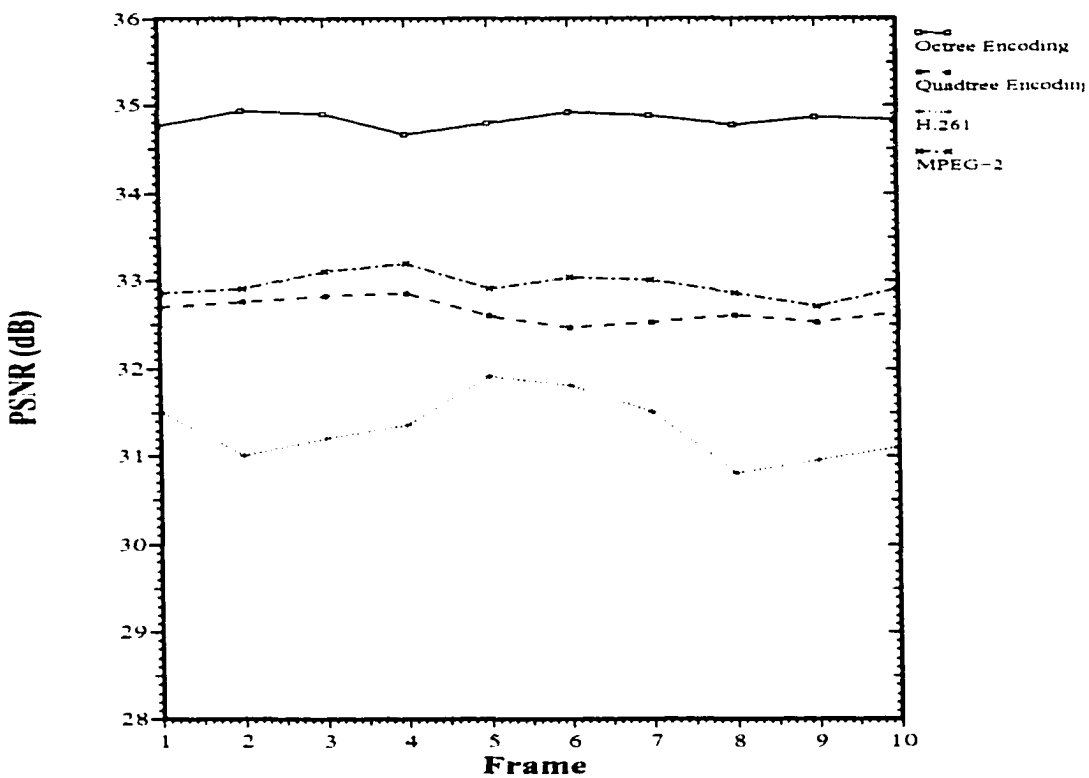


Fig. 5.8. PSNR for different encoding of the Miss America sequence.

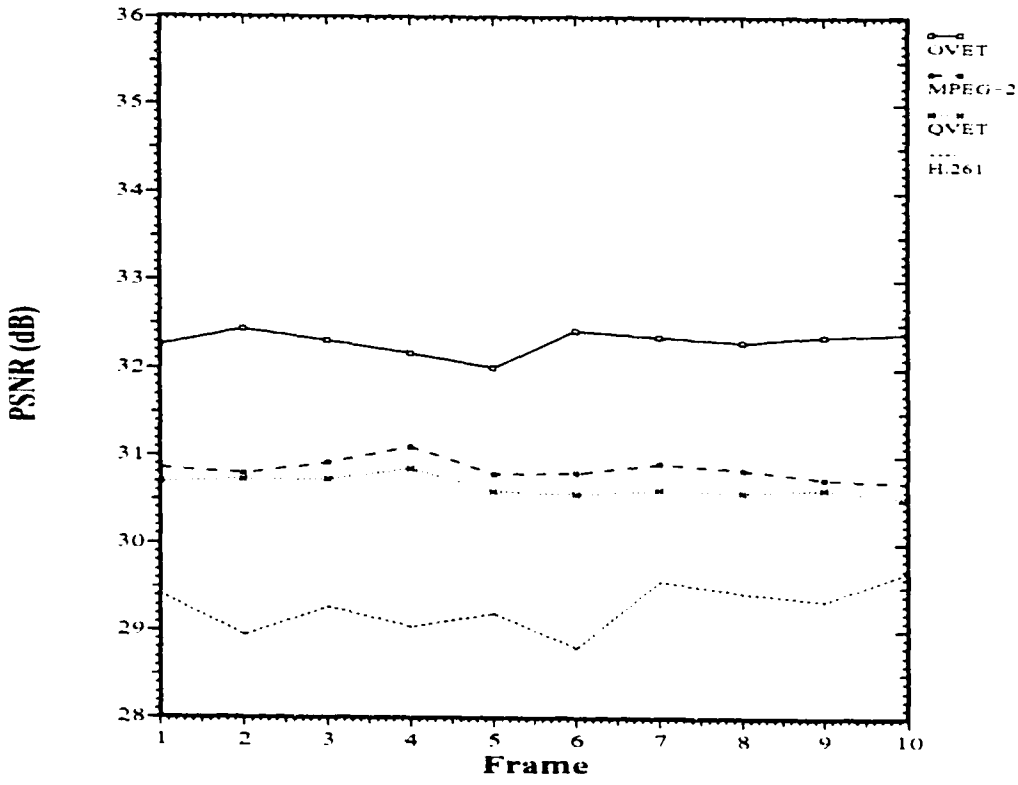


Fig. 5.9. PSNR for different encoding of the Football sequence.

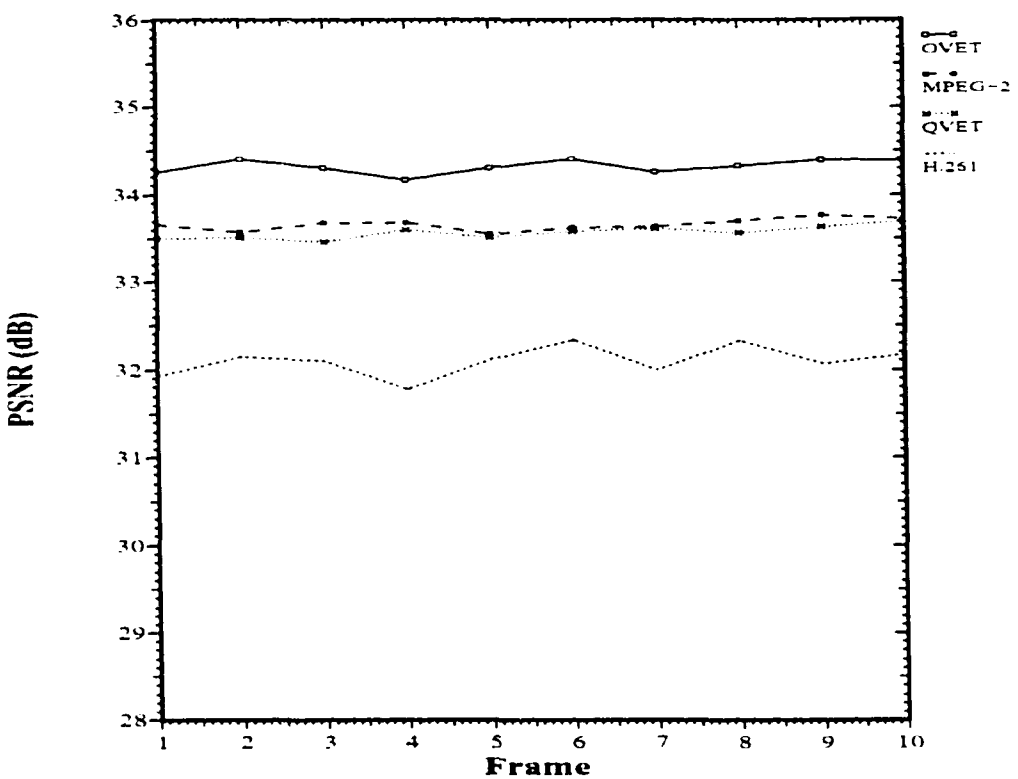


Fig. 5.10. PSNR for different encoding of the Table Tennis sequence .

5.3.2 Scalability

The ability of an encoding technique to be decoded at multiple levels of accuracy is very important in applications of computer communications. Our algorithm is scalable because all nodes of the octree are encoded, leaf nodes are encoded implicitly inside their parent non-terminal nodes. Most tree-based systems encode only the leaves of the tree to obtain a better compression ratio, while losing the scalability and robustness of the tree structure. To test the scalability of our new encoding technique, we performed the following experiment.

We encoded the first frame of the Miss America video (176x144) at bit rate 16 Kbps using our technique and MPEG-2, our encoding gave an image with PSNR 34.73 dB, while MPEG-2 gave an image with PSNR 32.85 dB. At the decoder, we scaled down both frames and plot the behavior of both. The experiment was repeated with several other frames with almost identical results.

Figure 5.11 shows the result of this experiment. The PSNR for the MPEG-2 encoding fell sharply when scaled down at the decoder, compared to our algorithm. For example, to scale down the frame size to 1Kbits (from 1.6 Kbits), our encoding gave a PSNR of 30.66 dB, while MPEG-2 encoding gave a PSNR of 27.51 dB. We conclude that to scale down a frame to a lower bit rate, our algorithm gives a better quality due to the naturally scalable structure of the octree.

5.3.3 Robustness

Figure 5.12 shows the reaction of OVET, QVET, and the MPEG-2 technique to random error loss across all levels of the octree. We encoded the first frame of the Miss America video (176x144) at bit rate 16 Kbps using the different techniques. OVET gave a frame with PSNR 32.70 dB, QVET gave a frame with PSNR 32.69 dB, while MPEG-2 gave a frame with PSNR 32.85 dB. Then we randomly dropped a certain percentage of data, and reconstructed the frame from the lossy data and calculated the new PSNR. As can be seen from the table, our representation is much

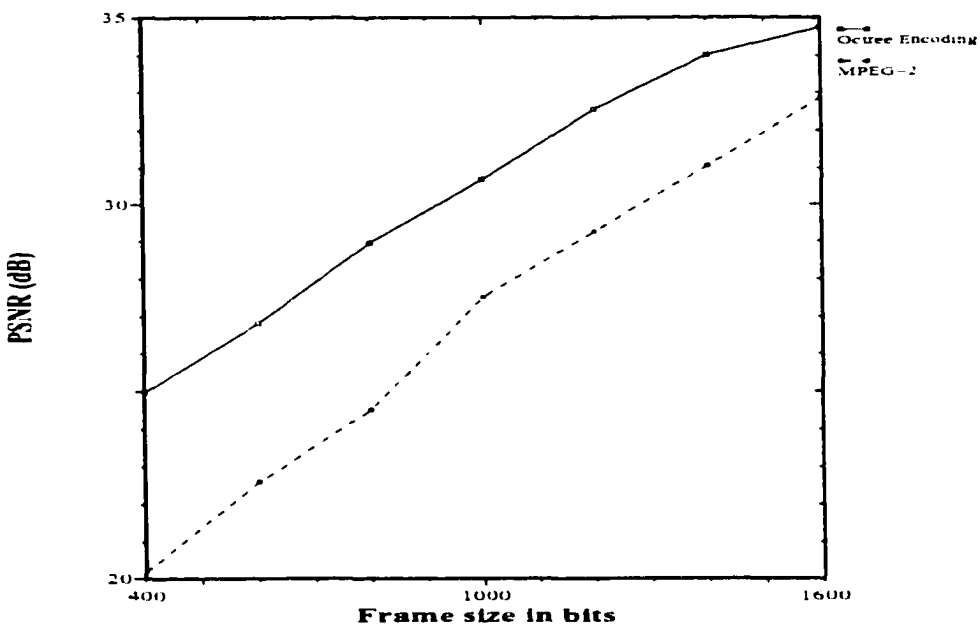


Fig. 5.11. Scalability results for the first frame of Miss America Sequence.

Loss	PSNR, OVET	PSNR, QVET	PSNR, MPEG-2
0%	32.70 dB	32.69 dB	32.85 dB
5%	32.47 dB	32.03 dB	30.67 dB
10%	32.07 dB	31.17 dB	28.14 dB
20%	30.91 dB	27.27 dB	23.07 dB
30%	29.56 dB	24.62 dB	20.83 dB
40%	28.62 dB	23.86 dB	18.47 dB

Fig. 5.12. Decrease in frame quality due to data loss.

more error resilient.

5.4 The Hybrid Video Encoding Technique, HVET

In this section, we describe our effort to combine both the quadtree and octree approaches to video encoding. The idea is as follows: First the video sequence is divided into GOPs where the size K must obey the function

$$K = 1 - 2^m \qquad where \quad m = 1, 2, 3, \ldots$$

The reason is that each GOP will be encoded into one quadtree and one octree. Figure 5.13 illustrates this abstraction for K = 5. In Section 5.2.1, we showed that, while working with OVET, a block size of (8x8x8) gave best results in all aspects. Therefore, we chose K to be 9, based upon our previous work.

The next step is to encode the first frame in the GOP as an independent I-frame using the quadtree-based image encoding technique, described in chapter 3. Then, for the rest of the GOP, we generate a "difference block", by subtracting the value of the pixels from the corresponding pixel in the I-frame. Then, the block is cut up into cubes (of size 8, if the GOP size is 9) and each cube is encoded separately

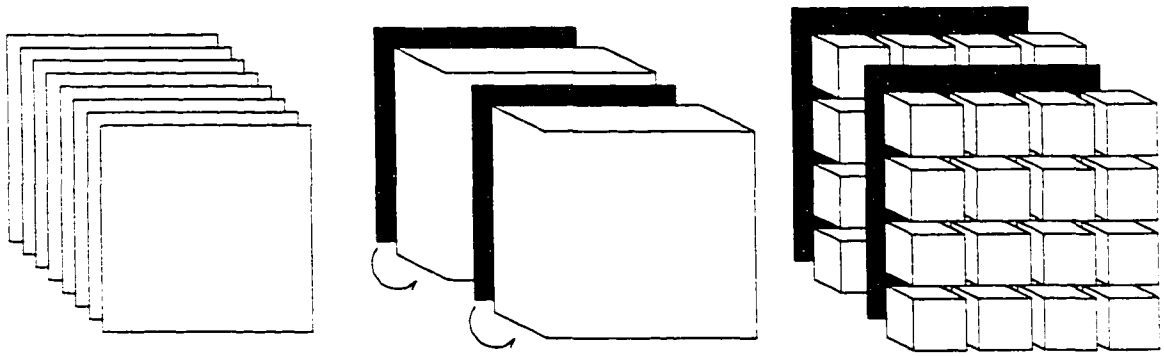


Fig. 5.13. Conversion from frames into I-frames and differential cubes.

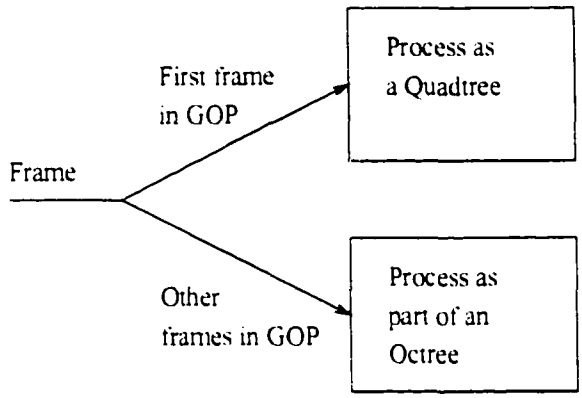


Fig. 5.14. Processing the different frames in the GOP.

in a manner similar to that described previously in this chapter. The idea is shown in figure 5.14.

The decoding process is the reverse of the encoding process. First, the I-frame is decoded using the decoding process of the quadtree-based image encoding technique. Then, we decode the differential octrees. We start by doing the reverse vector quantization to de-compress the octree structures. Then we convert from the octree structures to the 3D block. Finally, we add up the decoded I frame to each frame in the block to obtain the video sequence once again.

As we can see, the differential block is encoded and decoded as one entity, so

we need to be able to buffer frames at both the encoder and decoder. Also, we have to wait for all frames in the block to arrive before we can start building the octrees.

5.4.1 Experimental results

In this section, we describe how our new hybrid video encoding technique compares with MPEG2, QVET, and OVET. Our new hybrid technique shows great improvement over both the MPEG2 and QVET, and slightly better improvement over OVET. However, we must note that it is more time-consuming than OVET. In this section, we compare compression results, time complexity, and robustness. We do not discuss scalability since this encoding exhibits similar scalability properties as the octree-based encoding technique OVET, described earlier in this chapter.

Compression results

We used the same experiments, described in Section 4.2.1, used for testing the octree and quadtree techniques. The graph of figure 5.15 shows the results for all the different encoding for the MISSA sequence. The average bit rate is 16kbps at 10 frames/sec. From the chart, we find that our technique gives a better quality (for a fixed bit rate) than the other techniques. It has about 2.2 dB improvement over MPEG and 0.4 dB improvement over OVET. Figures 5.16 and 5.17 show the results of the same experiment with the football and table tennis test sequences, respectively.

Encoding Time Complexity

We now show that the cost of the hybrid technique, HVET is only slightly higher than that of the octree technique, OVET. From Appendix A, we see that the complexity of building the quadtree is of order 4/3 the size of the frame. The complexity of building the octree is 8/7 the size of the "differential block". Mathematical derivation for the complexity is explained in detail in [30].

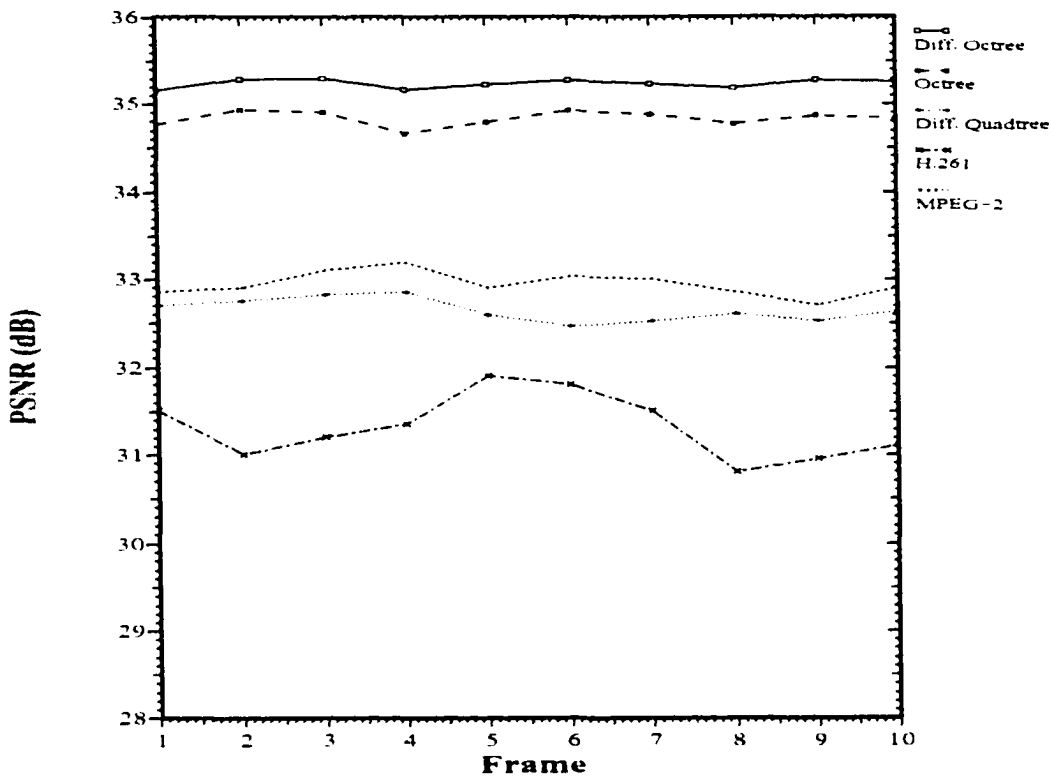


Fig. 5.15. MISSA compression results for HVET .

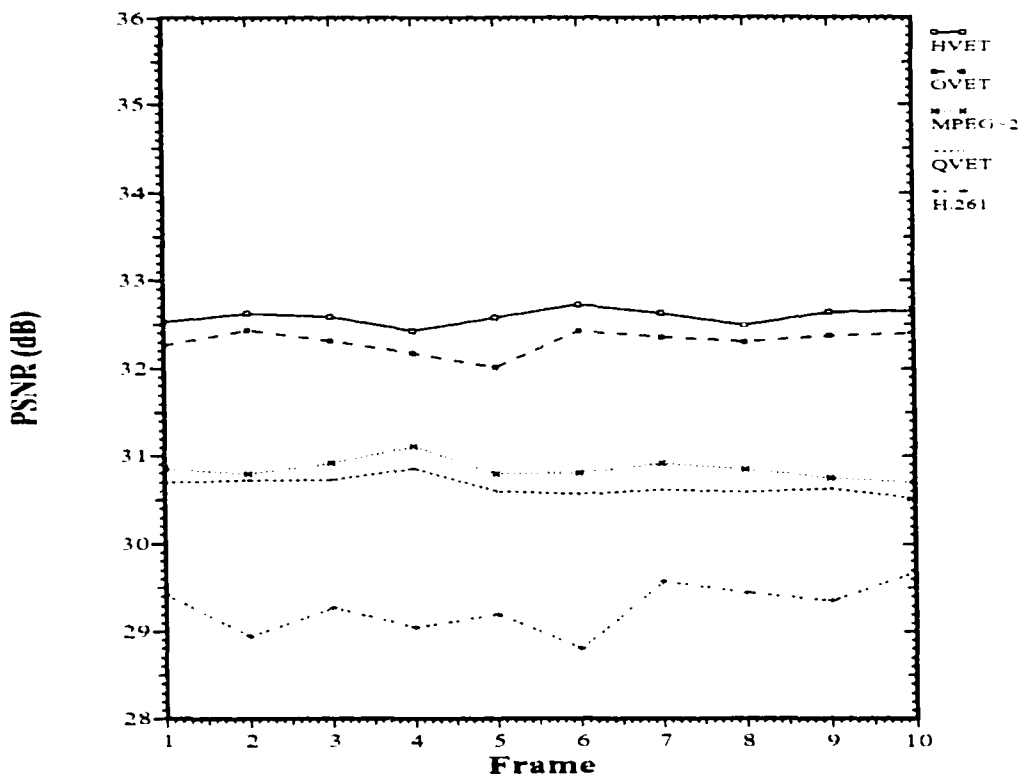


Fig. 5.16. Football compression results for HVET .

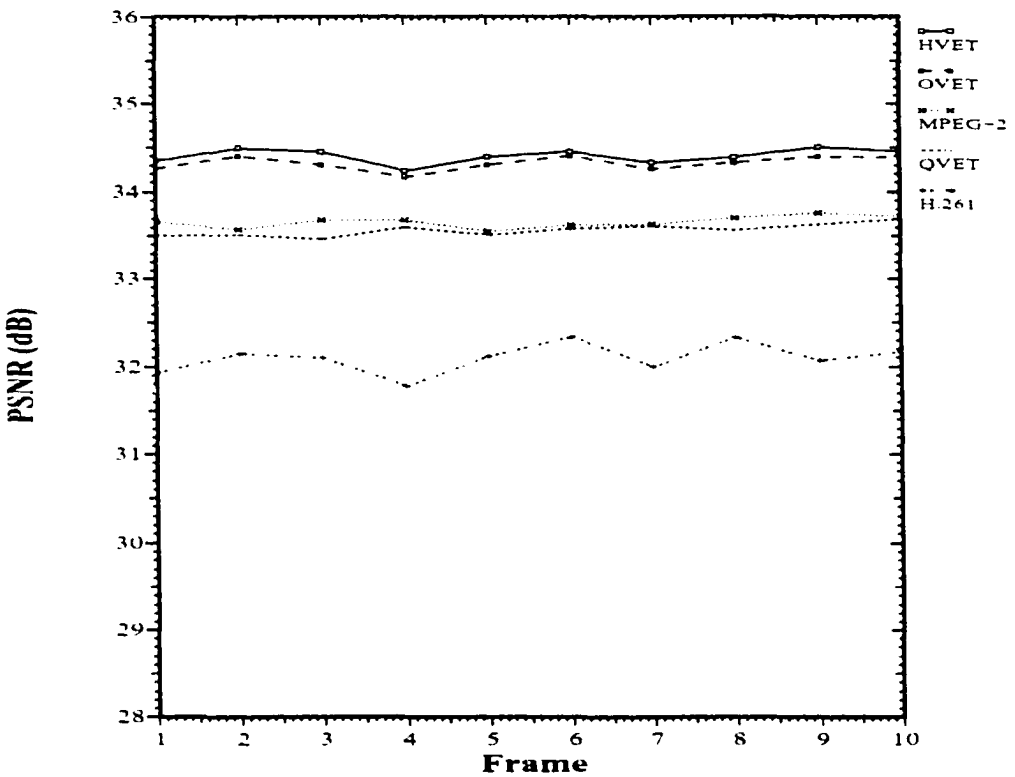


Fig. 5.17. Table Tennis compression results for HVET .

For a frame size of MxN pixels divided into (nxn) blocks, we calculate the actual size that will be encoded as:

$$S = (n \times n) \times \left\lceil \frac{M}{n} \right\rceil \times \left\lceil \frac{M}{n} \right\rceil bits$$

The complexity of QVET per frame can be stated as follows:

$$\begin{aligned} Complexity_{QVET} &= Complexity(Mean\ Removal) \\ &\quad + Complexity(Build\ Quadtree) \\ &\quad + Complexity(Vector\ Quan.) \\ &= O(S) + O(1.33S) + Comp_{VQ} \\ &= O(2.33\ S) + Comp_{VQ} \end{aligned}$$

The complexity of OVET with block size (nxnxn), per frame, can be stated as follows:

$$\begin{aligned} Complexity_{OVET} &= Complexity(Mean\ Removal) \\ &\quad + Complexity(Build\ Octree) \\ &\quad + Complexity(Vector\ Quan.) \\ &= \frac{O(S \cdot n)}{n} + \frac{O(1.1428S \cdot n)}{n} + \frac{Comp_{VQOCT}}{n} \\ &= O(2.1428\ S) + \frac{Comp_{VQOCT}}{n} \end{aligned}$$

The complexity of HVET with block size (nxnxn) and GOP size K, per frame, can be calculated as follows:

$$\begin{aligned} Complexity_{HVET} &= Complexity(one\ frame\ by\ quadtree) \\ &\quad + Complexity(n\ frames\ by\ octree) \\ &= \frac{(O(2.33 \times S) + Comp_{VQ}) + n(O(2.1428S) + \frac{Comp_{VQOCT}}{n})}{K} \end{aligned}$$

$$For\ K = 9$$

$$Complexity_{HVET} = O(2.1639\ S) + \frac{Comp_{VQ} + Comp_{VQOCT}}{9}$$

The complexity of the vector quantization for both the octree and quadtree, $Comp_{VQOCT}$

and $Comp_{V\text{-}Q}$, respectively, is dependent upon the size of the codebook we are searching, and the nature of the video sequence. Although performing vector quantization for an octree is more time consuming, from our experimentation we found out that it takes less half the time of the quadtree vector quantization, **per frame**. From the above equations, we conclude that the HVET takes only slightly more time to encode than OVET, for the same video sequence. We note that frequency based techniques, like MPEG2, are all of order the square of the block size for both encoding and decoding.

Robustness

To test for robustness, we used the same experiment used in the quadtree and octree encoding techniques. Figure 5.18 shows the reaction of our new hybrid representation, the octree-based representation, the quadtree-based video representation, and the MPEG-2 technique to random error loss across all levels of the octree.

We encoded the first frame of the Miss America video (176x144) at bit rate 16 Kbps using the different techniques. Then we randomly dropped a certain percentage of data, and reconstructed the frame from the lossy data and calculated the new PSNR. As can be seen from the table, our new hybrid video representation performs better than the MPEG and Quadtree representations. However, the octree technique still has the best robustness performance.

5.5 Summary

In this chapter, we described a new technique to encode video streams, using the octree representation method. Octrees are typically used in robotics and three-dimensional object representation. We proved that they also be used to successfully encode video as well, by viewing it as a sequence of three dimensional blocks. This method makes good use of both spatial and temporal redundancies. The compression results obtained were very good, giving about a 2 dB increase in quality. Also, we

Loss	HVET	OVET	QVET	MPEG-2
0%	32.73 dB	32.70 dB	32.69 dB	32.85 dB
5%	32.20 dB	32.47 dB	32.03 dB	30.67 dB
10%	31.50 dB	32.07 dB	31.17 dB	28.14 dB
20%	28.09 dB	30.91 dB	27.27 dB	23.07 dB
30%	26.74 dB	29.56 dB	24.62 dB	20.83 dB
40%	25.19 dB	28.62 dB	23.86 dB	18.47 dB

Fig. 5.18. Robustness experimental results for all encoding techniques.

obtained a good speed-up compared to the quadtree encoding technique.

Also, we developed a hybrid video encoding technique, HVET, by combining both the quadtree and octree methods. We encode the first frame in the GOP using the quadtree technique. Then, we use to generate a "differential block" from the other frames. And, we encode the block using the octree technique. This technique gave slightly better compression results than the octree technique but at the cost of an increase in encoding and decoding time, and slightly less error-resilience.

CHAPTER VI

CONCLUSION AND FUTURE EXTENSIONS

In this chapter, we summarize our motives and objectives, and the work we have done in this dissertation to achieve them. We describe the performance of our different techniques. We also present a list of possible future extensions to this work in the field of image and video communication.

6.1 Conclusion

In this dissertation, we present our ideas and efforts in the increasingly important field of image and video compression. A lot of researchers work in this field and new encoding techniques are being developed, as new applications and new requirements arise. Since visual data is too large to send over communication channels, compression has to be used. Traditional compression techniques are not suitable for communication applications, and therefore there is a large need for compression techniques tailored to that specific type of application.

A careful survey of the available encoding techniques showed that there was very few encoding techniques designed for interactive video-conferencing applications. Several techniques were developed for communicating video efficiently, but none covers the particular requirements of video conferencing. In a video conference, a session would typically have several participants, each of them encoding and sending one video stream, while receiving and decoding the streams send by the other

active participants simultaneously.

The requirements for video conferencing applications are speed of encoding and in particular speed of decoding, and a good compression ratio, because of its multi-stream nature. Another important requirement is scalability: the ability to decode a video stream at multiple levels of accuracy. A hierarchical encoding naturally is scalable. By having the data arranged in a hierarchy of multi-priority layers, receivers can drop data to scale down the quality without having to decode the entire frame first. Finally, robustness is another requirement, we expect some data to be lost during transmission, so the video stream should still be decodable even when parts of it are missing. Those requirements suggest the hierarchical quadtree structure as a suitable representation, as it satisfies the last two requirements without any additional effort.

Our design is based on the quadtree and octree representations. The quadtree is an approach to image representation based on successive subdivision of the image into quadrants. This process is represented by a tree of out-degree four. The octree is the three-dimensional equivalent of the quadtree, and is usually used to encode 3D objects.

First, we designed a new image encoding technique, QIET, based on the use of quadtrees and vector quantization. The algorithms divides the image into blocks, where the block size is dependent on the image size. Then, we apply a mean-removal algorithm on each block separately. Next, we generate the quadtree for each block. Finally, we use vector quantization to compress the quadtree data. Th resulting encoding is both highly scalable and robust. We compared our algorithm to other algorithms in the literature with respect to image compression, scalability and robustness. It performed well in all three aspects.

Our next step was to use QIET as a starting point for developing a differential video encoding technique, QVET. We extended it to encode video by applying the well-known IPB technique to the image encoding system. First, the sequence of frames is divided into blocks of K consecutive frames, called group of pictures (GOP).

The first frame of the GOP is chosen as a reference frame and all other frames are encoded based on the reference frame.

Next, we developed another, more efficient, method for encoding video based on QIET, called OVET, Octree-based video encoding technique. The idea was to encode each group of frames collectively as a three-dimensional object. We used octrees as our basic data structure, and vector quantization for compression.

Then, we combined the differential QVET technique and the octree technique, OVET, to generate a new hybrid video encoding technique, HVET. We encoded the first frame of each GOP independently using QIET. Then, the rest of the GOP was treated as a three-dimensional cuboid. We get the "differential" block by subtracting it from the independently coded frame. Then, we encode that block in a manner similar to OVET, conversion to octrees, then vector quantization.

The important issue is: how does our family of encoding techniques compare to others?. To verify its good performance, we tested our system using a set of standard benchmark video sequences. Our techniques are all of time complexity function of the frame size. Most encoding techniques are of time complexity order of square of the frame size. Also, our techniques are non-symmetric, so the decoding time is considerably less than the encoding time, a great advantage in multi-user systems.

We tested the compression performance of our techniques against the standard benchmarks, MPEG2 and H.261. QVET gave a reasonable performance, only slightly worse compression than MPEG2. But, OVET and HVET gave excellent results in compression, mainly due to the fact that they encode groups of frames collectively.

We also obtained good scalability and robustness results for all techniques, in comparison with MPEG2. This is mainly due to the flexible hierarchical nature of quadtrees and octrees.

In conclusion, we recommend the use of our octree-based techniques, OVET and HVET, as an efficient video encoding technique for multi-user heterogeneous

multimedia conferencing applications.

6.2 Future Extensions

The work we describe in this dissertation can be extended in the following ways:

- We mainly targeted the field of video conferencing when designing our encoding techniques. The requirements for that field are speed of encoding and in particular decoding, a good compression ratio, scalability, and robustness. A field that is similar in requirements is video-on-demand, typically used in digital video libraries [5, 27]. The field requires a fast way of decoding only. Encoding time can be excessive, since each video stream is encoded once and stored. A good compression ratio, scalability, and robustness are also important requirement. Our new quadtree and octree based techniques can be used in that field as well, with some modifications.

- An inherent problem in quadtrees and octrees is that they give a blocky effect when the frames are scaled down to obtain saving in size. Most frequency based techniques have the opposite problem, the frames get a fuzzy unclear effect when scaled down. There have been a lot of research in restoring scaled down images by smoothing down the borders between blocks in the quadtree [30]. For an interactive application like video conferencing, proper "restoration" of each frame is not practical. It would be useful to develop a method of quick restoration that can use previous frames in the video sequence to quickly restore current ones.

- We designed and implemented the encoding techniques for both image and video using a software implementation only. Most encoding techniques in actual use are implemented and built using hardware for greater speed. The next step in our technique would be to design and build hardware implementations

of our techniques and test them in real like applications. However, it is desirable to have the decoding technique available in both hardware and software, to minimize the cost for receivers.

- In this dissertation, we design different methods of video encoding, but we do not explore how our encoding technique will interact with the underlying communication backbone. A lot of research is being done in designing communication protocols that are suitable for multi-channeled scalable data [22]. Integrating our technique with such a system would be a very interesting future extension.

- Another way to extend the work in this dissertation is by further exploiting the nature of video conferencing applications. Typically the sequence is of a talking and gesturing person, similar to the standard benchmark "Miss America". The background information is completely irrelevant and should not take valuable communication bandwidth. A good extension to our work is to develop a background-removal system, that intelligently detects which parts of the frames are background and not encode it at all. This system should work in the background and not disturb the main encoding. It should analyze the frames as they are transmitted and judge which blocks are static and pure background, so they do not need encoding each time. The background-removal system should then periodically inform the main encoder system which blocks to completely ignore.

- Another possible extension is to integrate the fast quadtree-building technique developed by Samet [30] with our technique. Samet developed an algorithm to build a quadtree (or octree) in a progressive manner. So, instead of starting our quadtree-building with a full array of pixel, we build the quadtree as the data arrives, typically in a row-by-row fashion. This is particularly useful when building the octrees since each octree requires a large number of frames. The system is waiting idly until all the frames needed arrive. It would be more

efficient to start the work as soon as the frames arrive, even though it may take longer . We expect that the extra time needed to build the octrees will be less than the wait time for all frames to arrive. It will also save on the buffering requirements of the system.

- Another method of enhancement is to work on the end of sequence problem. We typically assume the video sequence is infinite. This problem really shows up if we have a large GOP size and an octree based encoding. The best solution would be to fill up the remaining data in the octree with dummy values, say all black, and encode it as a full octree.

REFERENCES

[1] CCITT. "Recommendation H.261: Video Codec for audiovisual services at p*64 kbps". 1990.

[2] A. Albanese, J. Blomer, J. Edmonds, and M. Luby. "Priority Encoding Transmission". The International Computer Science Institute, Berkeley, CA, TR-94-039, Aug. 1994.

[3] M. Blain and T. Fischer. "A comparison of VQ techniques". *EURASIP Image Commun.*, vol.3, no.1, Feb 1991.

[4] C. Chiu, and R. Baker. "Quad-tree product vector quantization of images". *Proc. SPIE Conf. Advances Image Compression Automat. Target Recogn.*, pp. 142-153, vol. 1099, March 1989.

[5] M. Christel, D. Winkler, and C. Taylor. "Multimedia abstraction for a digital video library". *Second ACM International Conference on Digital Libraries*, pp. 21-29, July 1997.

[6] R. Clarke and P. Cordell. "Coding of moving image sequences by recursive binary nesting". *Proc. Int. Picture Coding Symp.*, Italy, Paper 13.11, Sept. 1988.

[7] P. Cordell and R. Clarke. "Use of recursive binary nesting for coding moving image sequences". *Elect. Lett. 25*, pp. 362-363, 1989.

[8] P. Cordell and R. Clarke. "Block testing in a variable resolution spatially interpolative moving sequence coder", *SPIE-proc. 1360*, pp. 503-511, 1990.

[9] P. Cordell and R. Clarke. "Low bitrate image sequence coding using spatial decompression", *IEE proc. Part I comm. speech. vision*, 139, pp. 575-581, 1992.

[10] F. Denatale, G. Desoli, and D. Guisto, "Hierarchical image coding via mse-minimizing bilinear approx., *Elect. lett.* 27, pp. 2035-2036, 1991.

[11] A. Gersho, "On the structure of vector quantizers", *IEEE Trans. Inf. Theory*, IT-28, pp. 157-166, 1982.

[12] B. Girod, U. Horn, B. Belzer, "Scalable Video Coding for Multimedia Applications and Robust Transmission over Wireless Channels", *7th International Workshop on Packet Video*, Brisbane, Australia, March 1996.

[13] B.Girod, N. Faerber, E. Steinbach, "Standards Based Video Communication at Very Low Bit-rates", *Proc. VIII European Signal Processing Conference (EUSIPCO-96)*, Trieste, Italy, September 1996.

[14] B. Girod, E. Steinbach, and N. Farrber, "Comparison of the H.263 and H.261 Video Compression Standards", *Standards and Common Interfaces for Video Information Systems*, Critical reviews of optical science and technology, Volume CR60, pp. 233-251, October 1995.

[15] B. Haskell, A. Puri, and A. Netravali, *"Digital Video: An Introduction to MPEG-2"*, Chapman and Hall Publ., 1997.

[16] E. Kawaguchi and T. Endo, "On a method of binary picture representation and its application to data compression",*IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-2, pp. 27-35, Jan. 1980.

[17] B. Lamparter, and M. Kalfane, "The Implementation of PET", The International Computer Science Institute, Berekeley, CA, TR-95-047, Aug. 1995.

[18] B. Lamparter, A. Albanese, M. Kalfane, and M. Luby, "PET - Priority Encoding Transmission: A new, Robust and efficient video broadcast technology", The International Computer Science Institute, Berekeley, CA, TR-95-046, Aug. 1995.

[19] D. LeGall, "MPEG: A video Compression standard for multimedia applications", *Communications of the ACM*, vol. 34, pp. 47-58, April 1991.

[20] C. Leicher, "Hierarchical Encoding of MPEG Sequences Using Priority Ecoding Transmission (PET)", The International Computer Science Institute,

Berekeley, CA, TR-94-058, Nov. 1994.

[21] A. Lewis, and G. Knowles, "Video compression using 3D wavelet transform". *Elect. Lett.*, vol. 26, no. 6, pp. 396-398, 1990.

[22] K. Maly, H. Abdel-Wahab, C. Overstreet, C. Wild, A. Gupta, A. Youssef, E. Stoica and E. Al-Shaer, "Distance Learning and Training over Intranets". *IEEE Internet Computing*, Vol 1, No. 1, pp. 60-71, Jan. 1997.

[23] N. Nasrabadi, S. Lin, and Y. Feng, "Interframe hierarchical Vector Quantization". *Optical Engineering*, vol.28, pp. 717-725, July 1989.

[24] M. Oliver and N. Wiseman, "Operations on quadtree leaves and related image areas". *Computer Journal*, Vol. 26, no. 4, pp. 375-380, 1983.

[25] M. Oliver and N. Wiseman, "Operations on quadtree encoded images". *Computer Journal*, Vol. 26, no. 1, pp. 83-91, 1983.

[26] C. Podilchuk, N. Jayant, and N. Farvardin, "Three-dimensional subband coding of video". *IEEE Transactions on Image processing*, vol. 4 , no. 2, pp. 125-139, Feb. 1995.

[27] Y. Quintana, "Organization and retreival in a pictorial digital library". *Second ACM International Conference on Digital Libraries*, pp. 13-20, July 1997.

[28] N. Ranganathan, G. Romaniuk, and K. Namuduri, "A lossless image compression algorithm using variable block size segmentation". *IEEE Transactions on Image processing*, vol. 4, no. 10, pp. 1396-1405, Oct. 1995.

[29] H. Samet and C. Shaffer, "A Model for the analysis of neighbour finding in pointer-based quadtrees". *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. PAMI-7, pp. 717-730, Nov. 1985.

[30] H. Samet, "*Applications of spatial Data structures*", Addison-Wesley Publishing Company, June 1990.

[31] K. Sayood, "*Introduction to Data Compression*", Morgan Kaufmann Publishers, Inc., chp. 9, 1996.

[32] S. Selim and M. Ismail, "K-means type algorithms: a generalized convergence theorem and characterization of local optimality", *IEEE Trans. Patt. Anal.*

Machine Intell., vol. PAMI-6, pp. 81-87, 1984.

[33] S. Senbel and H. Abdel-Wahab, "Scalable and Robust Image Compression using Quadtrees", *Journal of Image Communication*, 1999.

[34] S. Senbel and H. Abdel-Wahab, "Octree-based Hierarchical Encoding for Video Conferencing", *IST/SPIE Symposium on Electronic Imaging Science and Technology*, San Jose, California, Jan. 1999.

[35] S. Senbel and H. Abdel-Wahab, "Scalable and Robust Differential Video Compression Using Quadtrees", *CS&I'98: Fourth International Conference on Computer Science and Informatics*, Oct. 1998.

[36] S. Senbel and H. Abdel-Wahab, "A Quadtree-based Image Encoding Scheme for Real-time Communication", *IEEE International Conference on Multimedia Computing and Systems*, June 1997, pp.143-150.

[37] S. Senbel and H. Abdel-Wahab, "A Scalable and Robust Image Encoding Technique for Real-time Communication", *CS&I'97: Third International Conference on Computer Science and Informatics*, pp. 191-194, March 1997.

[38] S. Senbel and M. Ismail, "Octree Generation From Object Silhouettes in Orthographic Views",*IST/SPIE Symposium on Electronic Imaging Science and Technology*, San Jose, California, Feb 1996.

[39] S. Senbel and M. Ismail, "The GrayCode Octree Representation Method", *IST/SPIE Symposium on Electronic Imaging Science and Technology*, San Jose, California, Feb 1996.

[40] E. Shusterman and M. Feder, "Image Compression via Improved Quadtree Decomposition Algorithms", *IEEE Transactions on Image Processing*, vol. 3, no. 2, pp. 207-215, March 1994.

[41] I. Stewart, "Quadtrees: storage and scan conversion", *Computer Journal*, Vol. 29, no. 1, pp. 60-75, 1986.

[42] P. Strobach, "Tree-structured scene adaptive coder",*IEEE Transactions on Communication*, vol. 38, pp. 477-486, April 1990.

[43] P. Strobach, "Quadtree-structured recursive plane decomposition coding of images", *IEEE Trans. Signal Processing*, vol. 39, pp. 1380-1397, June 1991.

[44] G. Sullivan and R. Baker, "Efficient Quadtree Coding of Images and Video",*IEEE Trans. Image Processing*, vol. 3, pp. 327-331, May 1994.

[45] M. Todd and R. Wilson, "An anisotropic multi-resolution image data compression algorithm".*ICASSP Proc.*, pp. 1969-1972, 1989.

[46] G. Wallace, " The JPEG still picture compression standard",*Communications of the ACM*, vol. 34, pp. 30-44, April 1991.

[47] R. Williams, "The Goblin Quadtree".*Computer Journal*, Vol. 31, no. 4, pp. 358-363, 1988.

[48] R. Wilson, M. Todd, and A. Calway, " Generalised quadtrees: a unified approach to to multiresolution image analysis and coding".*SPIE proc. 1360*, pp. 619-626, 1990.

[49] J. Woodwark, "The explicit quadtree as a structure for computer graphics".*Computer Journal*, Vol. 25, no. 2, pp. 235-240, 1982.

VITA

Samah A. Senbel was born in Alexandria, Egypt, on December 11th, 1970. She received her Bachelor of Science in Computer Science and Automatic Control from The Faculty of Engineering, Alexandria University, Egypt, in June 1992. She worked as a Teaching Assistant for the Department of Computer Science at The Arab Academy for Science and Technology from September 1992 to December 1994. In December 1994, she received her Master of Science degree from the Department of Computer Science at The Faculty of Engineering, Alexandria University, Egypt. She started working on her Ph.D. Degree in Computer Science at Old Dominion University, Virginia, in January 1995. During the course of her Ph.D. work, she co-authored eight scientific papers and technical reports. She is a recipient of the GAANN fellowship since 1997.

Permanent address: Department of Computer Science
Old Dominion University
Norfolk, VA 23529
USA

This dissertation was typeset using LaTeX* by the author.

*LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.